(4)

RADC-TR-88-324, Vol VII (of nine), Part B
Interim Report
March 1989

NORTHEAST ARTIFICIAL INTELLIGENCE CONSORTIUM ANNUAL REPORT 1987 Parallel, Structural, and Optimal Techniques in Vision

Syracuse University

Christopher M. Brown

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

This report has been reviewed by the RADC Public Affairs Division (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-88-324, Vol VII (of nine), Part B has been reviewed and is approved for publication.

APPROVED: Michael D Richard

MICHAEL D. RICHARD, Capt, USAF
Project Engineer

APPROVED: Walter J. Senus

WALTER J. SENUS
Technical Director
Directorate of Intelligence & Reconnaissance

FOR THE COMMANDER: Igor G. Plonisch

IGOR G. PLONISCH
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (IRRE) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific doucment require that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS N/A
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) N/A	5. MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TR-88-324, Vol VII (of nine), Part B
6a. NAME OF PERFORMING ORGANIZATION Northeast Artificial Intelligence Consortium (NAIC) 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (IRRE)
6c. ADDRESS (City, State, and ZIP Code) 409 Link Hall Syracuse University Syracuse NY 13244-1240	7b. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Rome Air Development Center 8b. OFFICE SYMBOL (If applicable) COES	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F30602-85-C-0008
8c. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 62702F; PROJECT NO. 4594; TASK NO 18; WORK UNIT ACCESSION NO E2
11. TITLE (Include Security Classification) NORTHEAST ARTIFICIAL INTELLIGENCE CONSORTIUM ANNUAL REPORT, 1987, Parallel, Structural, and Optimal Techniques in Vision	
12. PERSONAL AUTHOR(S) Christopher M. Brown	
13a. TYPE OF REPORT Interim 13b. TIME COVERED FROM Dec 86 to Dec 87	14. DATE OF REPORT (Year, Month, Day) March 1989 15. PAGE COUNT 192
16. SUPPLEMENTARY NOTATION This effort was performed as a subcontract by the University of Rochester to Syracuse University, Office of Sponsored Programs.	

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Artificial Intelligence, Temporal Reasoning, Planning, Plan Recognition, Computer Vision, Parallel Computation, Multiprocessing, Edge Detection
12	05		
12	07		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The Northeast Artificial Intelligence Consortium (NAIC) was created by the Air Force Systems Command, Rome Air Development Center, and the Office of Scientific Research. Its purpose is to conduct pertinent research in artificial intelligence and to perform activities ancillary to this research. This report describes progress that has been made in the third year of the existence of the NAIC on the technical research tasks undertaken at the member universities. The topics covered in general are: versatile expert system for equipment maintenance, distributed AI for communications system control, automatic photointerpretation, time-oriented problem solving, speech understanding systems, knowledge base maintenance, hardware architectures for very large systems, knowledge-based reasoning and planning, and a knowledge acquisition, assistance, and explanation system. The specific topic for Part A of this volume is a model theory and axiomatization of a logic for reasoning about planning in domains of concurrent actions. Part B addresses various aspects of parallel, structural, and optimal techniques in computer vision.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL MICHAEL D. RICHARD, Capt, USAF	22b. TELEPHONE (Include Area Code) (315) 330-7788 22c. OFFICE SYMBOL RADC/IRRE

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

7B Parallel Computer Vision

Report submitted by:
Christopher M. Brown

Computer Science Department
University of Rochester
Rochester, NY 14627

TABLE OF CONTENTS

1. Summary

The vision group at Rochester has spent the year investigating various aspects of parallel computer vision, with the goal of building behaving, real-time systems that perform multi-sensory integration. What sets our current work apart is a commitment to the idea that an intimate coupling of sensory and motor capabilities is a way to make progress on the vision problem. Behaving animals have such a coupling, and its benefits have been demonstrated analytically in several papers from Rochester and elsewhere. This year has shown progress in building hardware and software to implement the theory.

Hardware Developments: We have acquired special-purpose parallel pipelined hardware for low-level vision, and have upgraded our 16-processor Butterfly Parallel Processor with faster CPUs and floating point hardware. The Butterfly is now connected with the rest of the vision hardware (the pipelined device and a fast Sun/3) via the VME bus.

Individual Activities: David Sher completed his Ph.D. and is now at the State University of New York at Buffalo. He and Chris Brown are continuing collaborative work. Paul Chou is continuing his extension of Sher's probability-based feature detection, and will finish his thesis in 1988. Paul Cooper is exploring theoretical properties of his parallel object recognition algorithm. Bob Potter is implementing Chou's evidence-combination algorithm on the Butterfly. Chris Brown and Nancy Watts have been working on principal view computations in the domains of solid planar polyhedra and the "blobs and sticks" domain of Paul Cooper.

In summary, our work this year broadened its focus from the optimal selection of feature detectors in a Bayesian framework. These results were used this year and continue to be used in a working system that applies a Markov Random Field formulation of the segmentation (objecthood recognition, figure-ground separation) problem. Further, this year our work moved in the direction of acquiring and commissioning real-time vision hardware and using the hardware in applications. Theoretical work continues to be important, and this year we made progress on the theory of principal views and convergence properties of two sorts of parallel networks: connectionist nets for recognition and Markov Random Field models of segmentation.

2. Active Vision

Recent technological and theoretical developments have made possible the active, real-time control of robotic visual sensors as well as the real-time processing of their output. At the University of Rochester, we are starting a serious research effort involving the analysis, design, construction, and application of active sensing. The major scientific premise is that *an intimate cooperation between the sensory and motor systems of a behaving, sensing entity allows significant improvement in the quality and usability of visual information.*

The potential of such cooperation has long been recognized but until recently has not been technically practical. At Rochester, we have several state-of-the-art devices that we should like to integrate into a coherent hierarchical hardware and software organization, ranging from a 16-processor M68020/68881 Butterfly down to a MaxVideo pipelined image processor, a 3-degree-of-freedom binocular robot head, and (in the near future) a PUMA 761 robot body. Progress this year has yielded some hardware, some software, several committed people, and certain encouraging results.

One of our first goals was (and still is) to develop a set of basic capabilities (reflexes, low-level skills) upon which active vision can be securely based. Foremost among these is *fixation*, or *foveation*. Closely related to foveation is *vergence*, which can be useful for stereo fusion. Dynamic behavior built on fixation and vergence includes a *VOR*-like reflex which maintains fixation despite commanded head movements. *Tracking* is a dynamic version of fixation, used both to inspect moving objects and to inspect objects while the head is in motion. Last, *saccades* are rapid eye movements that have several important properties: visual processing ceases during them, they account for both motion and position of the target object, they interact in a smooth way with other capabilities, and they are used to build up a spatial representation of the environment. *Develop a control structure that integrates the basic skills coherently.* We can learn from primate systems [Cornog 1986; Ballard 1987] much about the form successful control systems take. For example, in the human system the VOR is basic, but is overridden by the tracking system, which can in turn be interrupted by the saccadic system. The capabilities are associated with basically different modes of operation that must cooperate--for instance, optic flow is not interpreted as a moving surface if it arises as the background of a tracked object. The basic control structure must support perceptual tasks that have varying demands for resources on the system, as well as provide a basis for a hierarchy of plans and skills.

This year we have produced demonstrations of saccades, tracking, and vergence using our "robot head," and we have explored control structures for real-time following of several moving objects at once [Ballard et al. 1987]. Figures 1 and 2 show our vision laboratory configuration and the binocular robot head.

2.1 Robot Head and Pipelined Image Processor Utilities

Both our parallel computers are complex pieces of hardware. We have hired a full-time vision programmer who is working on the systems and utilities aspects of both the MaxVideo and Butterfly systems.

The first project is to investigate how to get programs running on the Butterfly to communicate with the MaxVideo hardware over the VME bus. This will enable us to have multiple processes running on the Butterfly that control different operations on the MaxVideo system.

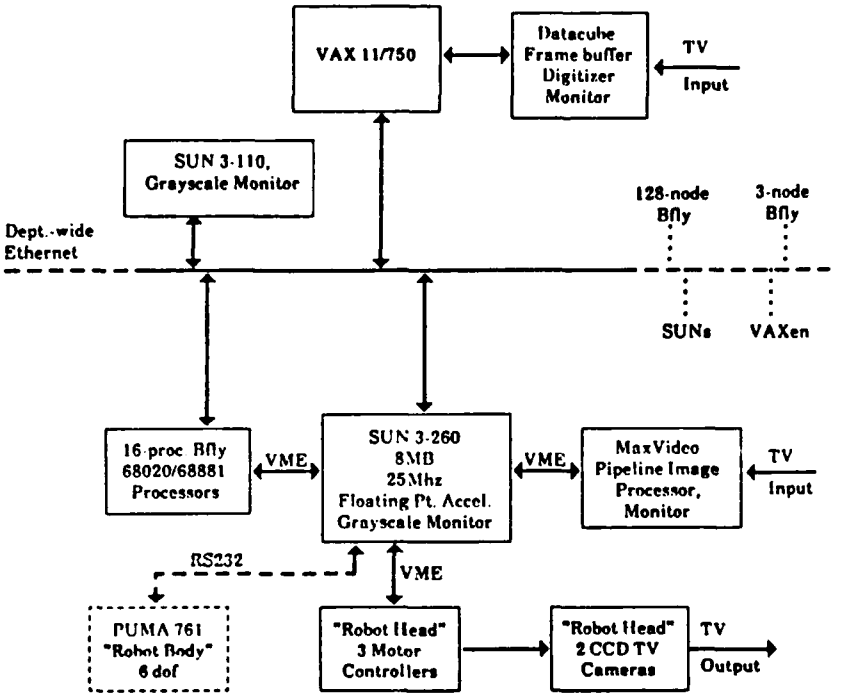


Figure 1: Vision and Robotics Laboratory Hardware

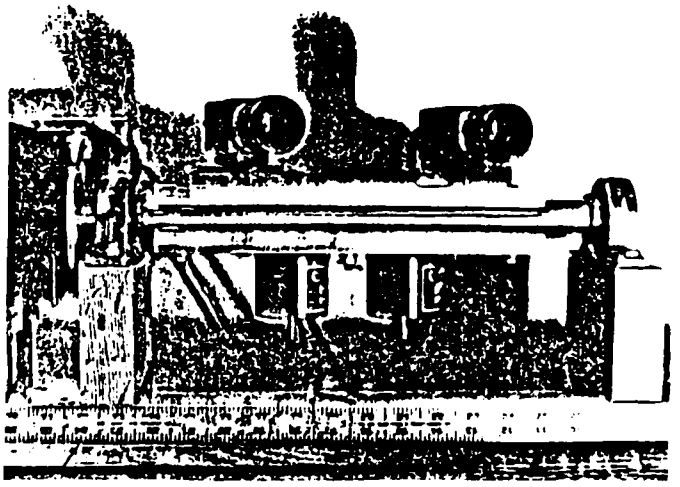


Figure 2: The Robot Head

Frequently, the output of a hardware package will appear as mysterious as the "black box" which created it. Therefore, in order to take full advantage of the capacity of an architectural design, it is first necessary to comprehend the reactions to basic inputs.

With this intent, the capabilities of the VFIRII board were explored. An examination of actual bit patterns under a variety of test circumstances helped to answer questions concerning the nature of previous outputs, while it broadened our understanding of the designer's goal in adding various special features.

In particular, the linear nature of convolution was exploited to create arbitrarily-sized templates by either cascading VFIRII boards or recirculating partial products using the onboard delay compensation. Possible future applications of this feature include large pattern recognition, differentiation, and faster processing times.

2.2 Spatio-temporal Feature Detectors

Another project is to investigate motion detection through the use of temporal difference operators. The goal is to have the two robot eyes track a randomly shaped and randomly patterned object as it moves across a randomly patterned background. Normally if the object is not in motion then the eyes would not detect the object due to its random pattern. However, when the object is in motion the temporal difference operator will detect the leading and trailing edges of the object. This project would then be integrated with the Rover project [Coombs and Marsh 1987; Ballard et al. 1987].

2.3 The Saccadic System

We have used portions of the MaxVideo family of hardware to implement an algorithm to track a gray-level-distinguishable object with a moving camera (Figure 2), and another algorithm to identify the gray-level-distinguishable objects in the field of view of the camera and make the camera perform a saccadic motion to one of them.

Further work, as part of a Master's thesis to derive depth information about objects in the field of view by using information derived from images obtained from a moving camera fixated on a point in the field of view, remains to be done. It awaits the availability of software to control the Signal Processing board in the MaxVideo hardware.

2.4 Stable Viewing and Multi-View Integration

One pilot project we completed was a view de-jitterer; it explored image registration using projections for real-time computability. The task is to align an image at time t with one from time t-1 so that any shifts of the camera are removed. This is useful in many areas: in active vision, it can steady the picture for input to higher-level routines; in large-scale mapping it can align overlapping pictures to make a single larger picture with no noticeable joining line; and in a device such as electronic binoculars or video cameras it can remove the jitter which is due to their being hand-held.

The implementation had the further goal of working in real-time on the MaxVideo hardware. To that end, the algorithm of using a row and a column projection rather than the whole picture was used, reducing the computation at the convolution step by hundreds of times. A first-cut version running on the Sun worked fine, at a rate of about two frames per second, so it looked promising to speed that up to 30 per second by using the MaxVideo hardware.

The underlying algorithm is essentially well-known in the literature, but was an encouraging implementational exercise and yielded some important information about our hardware.

2.5 Some Technical Details: by Steve Colwell

One paradigm for use of MaxVideo is simply as a frame grabber, reading the frames into the Sun, and doing all the processing inside the Sun. This is in some sense easier than using the MaxVideo hardware, mainly because the debugging and development tools on the Sun are better than what is available when doing algorithm development directly with the datacube boards. The speed that ought to be achievable for this task is about .5 microseconds per 32 bit transfer, which is 8 megabytes/second, which is 32 512x512 frames per second.

Our first results were very slow as a result of using the pixel-by-pixel movement instructions. They got better when we used the row-by-row block move routines, and even better when we abandoned the supplied routines and did the moves directly from the memory-mapped VME bus address to the screen memory of the Sun. However, even for this last case we only got three frames per second.

After many experiments and conversations with the designer of the board at Datacube, we determined that it wasn't going to get any better, due to a combination of two problems. The first problem is that the MaxBox that we have doesn't have the 32 bit VME bus option. This doesn't matter with most other datacube boards, since they are only capable of 16 bit transfers anyway. In the case of the ROIStore board, though, we could get twice as much transfer speed (up to 6 frames per second, quite an improvement) by ordering the P2 bus option for the MaxBox.

The second problem is more technical. The cycle times we see on accesses to the ROIStore are around 1700 ns, which is longer than is reasonable. The Sun itself only takes 250 ns. That means the ROIStore is taking the rest; but how? The ROIStore board is capable of a cycle speed of 800 ns. This is implemented by splitting up each 800-ns cycle into three equal slices, one of which is the VME bus cycle. If the request from the Sun occurs at exactly the right moment, just when the ROIStore is ready to service the Sun slice of the 800-ns cycle, the response of the ROIStore can be as fast as 450 ns. However, if the Sun request comes just after its slice has gone by, the Sun has to wait $800+450 = 1250$ ns. Even this worst-case time, which is pretty horrendous as these things go, is not as bad as what we see, and we should be seeing

what the datacube engineer calls the "cycle" time of 800 ns anyway, not the worst case time.

The reason for the extra delay is a classic example of non-communication between design engineers. The interface board between the Sun and the MaxBox takes 150 ns or so of delay, which is normally pretty fast. However, in this case, the effect is to make the length of a move instruction be, in the best case, 450 ns (minimum time from the ROIStore) + 150 ns (interface board) + 250 ns (Sun time) = 850 ns. The next move instruction, which comes immediately afterward, has to wait for the next cycle to come around, a long 750 ns, so the total cycle time is 850 ns (actual reasonable delay) + 750 ns (designer obtusity delay) = 1600 ns. Obviously, the main thing is that we shouldn't expect to do ROIStore-to-Sun accesses in anything like real time (most people wouldn't anyway). For those of us who still prefer to load things into the Sun to do algorithm design and such, the top speed from the ROIStore is three frames per second. Of course, there are some tricks left.

First, we can order the P2 bus option for our MaxBox. This is a cheap way to double the transfer speed by doing one 32-bit transfer every 1600 ns instead of one 16-bit transfer each 1600 ns. Second, we could load smaller pictures than 512x512. We have random access to any part of the image that is desired, so just taking the middle 256x256 part of the screen would be four times faster, or twelve frames per second. We don't really get 512x512 pixels anyway, since a border on the left and top of the picture of something less than 50 pixels is left blank. Third, we could read through a different device than the ROIStore. The Euclid board, our signal processing computer board in MaxVideo, has memory with a 150-ns cycle time rather than 800 ns, so accesses to it (although restricted to 16 bits) can occur about twice as fast. That means that it would be about the same as the ROIStore performance would be with the 32-bit bus. We have now begun to run applications on the Euclid board, so this option is a realistic one.

In the process of learning all this, we developed some general techniques. A picture can be put up on the Sun screen in 40 ms instead of the 200 ms it takes using the block moves that Sun provides. Any size frame can be read directly from the ROIStore as one routine instead of using the block moves that datacube provides. A value mapping table which converts values from the 0-255 range of the data cube to a logarithmic 0-127 range for display on the Sun is available, along with the code to load it into the DigiMax hardware so no cpu time need be used to implement the table lookups.

3. Parallel Object Recognition: by Paul Cooper

This year I completed the implementation of my stereo algorithm. My main activity has been in the formal analysis of my connectionist parallel object recognition algorithm.

3.1 Related Work

Early in the year (during April and May) I completed my work on stereo from structure. This work is available as TR 216, "Order and Structure in Stereo Correspondence" [Cooper 1987a], which has been submitted to the International Journal of Computer Vision.

During the spring I also built a utility which will print grey-scale images on our laser printers, using half-toning. The utility also allows such images to be included in troff and TeX documents (such as TR 216). During the early summer I attempted to develop a coherent framework for uncertainty in the representation and recognition of structure. Unfortunately, this endeavour was overly ambitious, and had to be abandoned.

Instead, I began to focus my efforts in two directions: object-model matching with uncertainty, and using domain knowledge explicitly to constrain the recognition indexing problem. To investigate the use of domain knowledge, I developed a grammar which describes all allowable tinkertoy configurations that yield models of animals. In future work, this grammar will be used to assist in controlling the recognition process. Figure 3 shows example domain objects, early visual processing, and the catalog of possible objects.

My main task in the fall has been analyzing and proving the computational characteristics of the connectionist network we developed in earlier work, which does graph matching in parallel [Cooper and Hollbach 1987]. A paper that describes this work was recently completed [Cooper 1987b], and is now described briefly.

3.2 Structure Recognition by Connectionist Relaxation

In summary, the paper describes a connectionist implementation of discrete relaxation, for labeled graph matching. The application is fast parallel indexing from structure descriptions. Complexity considerations limit the network to the detection and propagation of unary and binary consistency constraints.

The convergence of the algorithm is proved. The desired behavior of the algorithm is formally specified, and the fact that the network correctly computes this is proved. Explicit and exact space and time resource requirements are developed.

The general utility of discrete relaxation is well known, and the abstract form of the algorithm is well understood. But the specifics of the connectionist implementation illuminate some interesting issues.

First, with the connectionist implementation the complexity of the algorithm can be made very precise. The exact space and time requirements of the algorithm have been specified. The explicit nature of the implementation makes extending the algorithm in parallel trivial, and its resource requirements remain explicit and easy

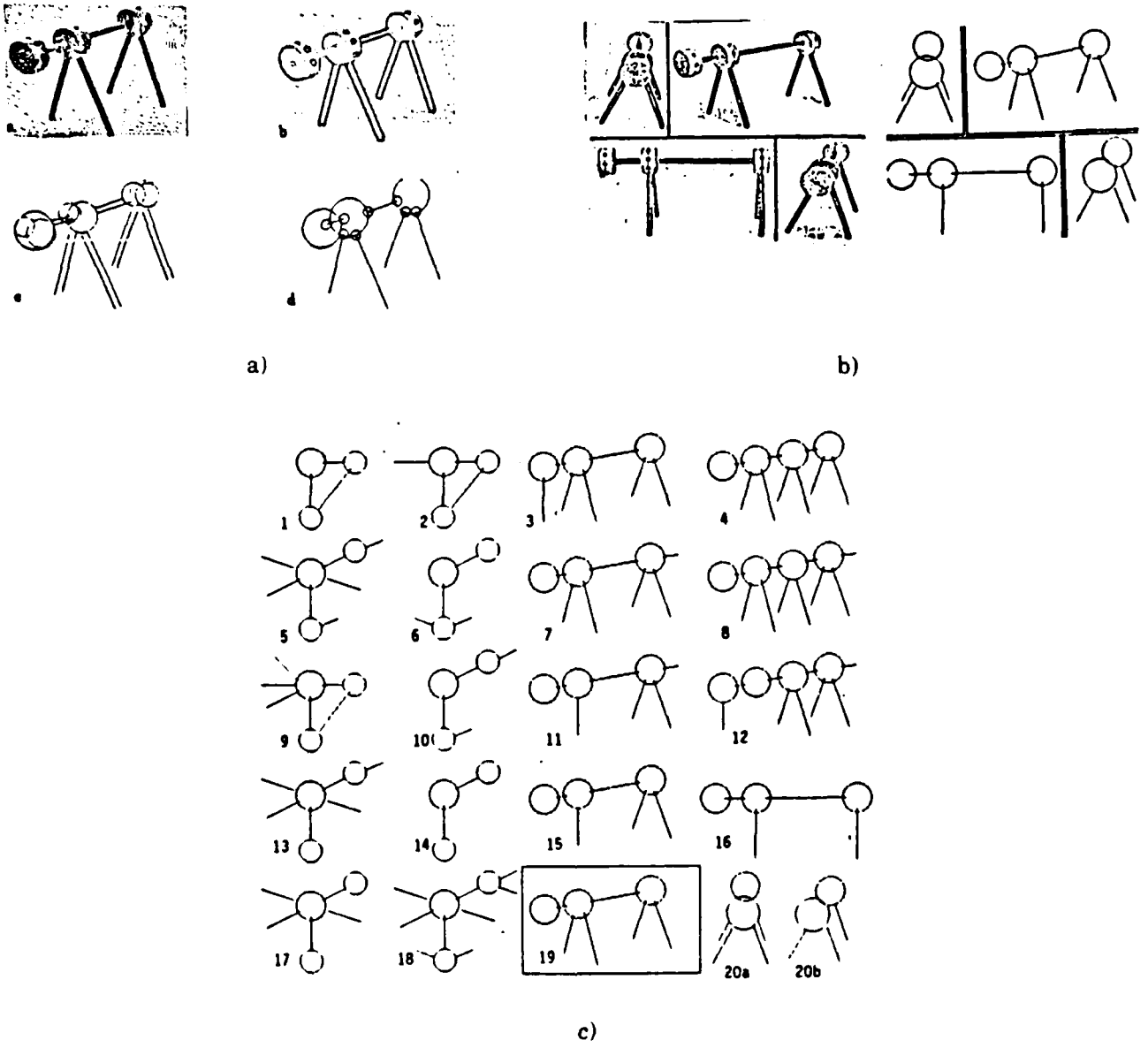


Figure 3: (a) Processing steps for tinkertoy objects; (b) input and processed tinkertoy objects; and (c) the catalog of principal views.

to compute. The theoretical speed one expects to obtain with massively parallel implementations of algorithms is provably present. With many implementations of relaxation such issues are not strongly addressed.

Second, it is interesting to observe how working within the connectionist paradigm constrains a design. The requirement that there be no interpreter means each relevant value must be represented by a unit. This suggests, in the context of the graph matching problem, that representing matchings for ternary or higher relations would become prohibitively expensive. Furthermore, bandwidth

limitations on the messages the units can send each other (inspired by neurual communication bandwidth limitations) restrict the character of what can be computed with unary and binary constraints. For example, it would be difficult to modify the algorithm so that more than one unary predicate could be used (like [Kitchen and Rosenfeld 1979] do). Likewise, bandwidth limitations restrict the potential utility of the unary predicate itself. Interestingly, the overall resource requirements of the algorithm suggest an upper bound on the number of parts a structure representation can have before a more efficient representation such as a hierarchy is required. (Structure representations with more than a small constant number of parts, like in the small double digits, would become unwieldy)

The paper contains somewhat stronger and more specific results about what can be matched than are typically found. These results were obtainable because the exact nature of the algorithm was very tightly constrained (e.g., only unary and binary constraints, with propagation occurring only under very specific conditions).

The particular performance characteristics of the algorithm suit the task problem--fast indexing from structure descriptions--very well. Complete recognition (with verification) requires that a complete correspondence between object and model be established, and this algorithm is unable to do so. But it can be used to filter out large numbers of candidates quickly for recognition, so later more expensive processes can be more confidently and efficiently applied.

Finally, the implementation and proof provide a basis from which to investigate more general indexing problems. In future work I plan to investigate indexing from structure with uncertain and inexact structure descriptions. Connectionist implementations seem to be natural hosts for such problems.

4. Principal View Geometry

Our work in representation of objects for recognition has yielded the following results so far.

1) *Theoretical and algorithmic results for principal view calculations on planar polyhedra*, including an algorithm to generate all spatial volumes from which different principal views are seen [Watts 1987].

2) *Theoretical and algorithmic results for principal view recognition using connectionist networks and objects characterized by their structure* (tinkertoy objects) [Cooper and Hollbach 1987; Cooper 1987b]. Figure 4 shows some images used by the system and its catalog of potential views.

3) *Theoretical and algorithmic results for the calculation of principal views for tinkertoy, pure blob, and pure stick objects.* One result is that there are at most $O(n^4)$ different views (where view equivalence is defined by the overlaps of individual blobs or sticks in the projection) on any viewing sphere (i.e., for orthographic projection, or for perspective projection of any focal length). This

result is consistent with the analogous result for convex polyhedra. We have no analogous result for the total number of principal view regions in all of viewing space (this is known to be $(n^3-5n)/6$ for convex polyhedra, but it seems likely that the analogous n^6 bound will hold up.

The algorithmic part of the work creates the views themselves, and for each equivalent view, gives the approximate fraction of the viewing sphere that yields the view. The program works by recursively splitting facets of a viewing polyhedron if views from the facet corners differ. Results can thus be obtained for any level of resolution. Analytic computation of the viewing volumes that yield equivalent views, as in [Watts 1987], has proved prohibitively complex. View equivalence is flexibly described as the equivalence of any combination of stick-stick, blob-stick, and blob-blob incidence matrices. Thus views may be generated for a view catalog, and rated in quantitative order of likelihood. This quantification will be useful in planned future work in which strategies for recognition are precompiled based on information theoretic criteria [Swain 1987]. Figure 4 shows some output from the principal view analyzer.

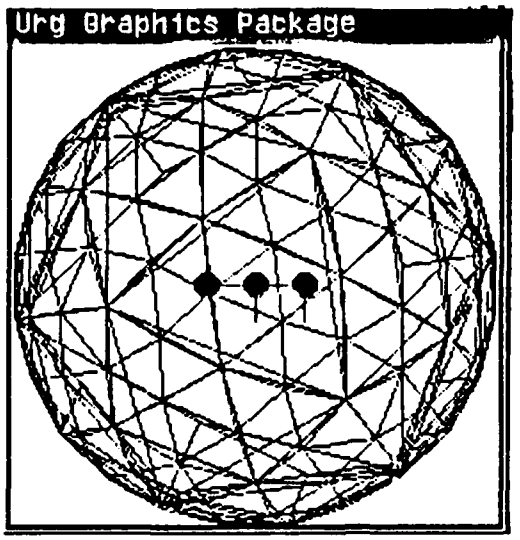
Our work has so far determined that though there are a large number of "different" views, there may be a high variance in their probability. For the horse of Figure 4, there are 35 different views at medium resolution, but one (no blob or stick adjacencies) accounts for 63% of the viewing sphere, with none of the other 34 views accounting for more than 4%. The view catalog will be useful for creating catalogs as in Figure 3. The quantification of probabilities will be useful in planned future work in which strategies for recognition are precompiled based on information theoretic criteria [Swain 1987]. However, this "probability" figure seems to have little to say about qualitative appearance differences, saliance, or information content of the view.

5. Markov Random Fields for Evidence Combination

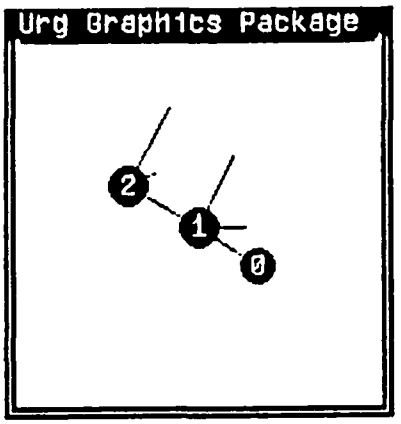
Paul Chou and Rajeev Raman have recently developed a new algorithm for image segmentation [Chou and Raman, 1987]. A Markov Random Field (MRF) is used to represent observational data and a priori knowledge about image edges. The new Highest Confidence First (HCF) algorithm approximates the minimum energy labeling of the MRF, producing an interpretation of the positions of edges that is nearly optimal under the Maximum A Posteriori (MAP) criterion. The solutions constructed compare favorably with ones produced by pre-existing methods and the computation is more predictable and less expensive.

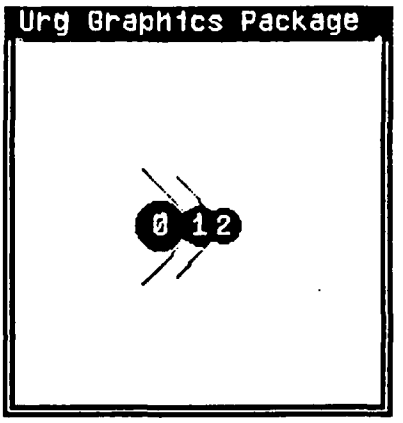
The solutions to many computer vision problems can be formulated as the minimum energy states of thermal dynamic systems. However, the complexity of the energy functions prevents the analytic solution of the minimization problems. Figure 5 illustrates the use of the new method, the HCF algorithm, for approximating the optimal solution to the image segmentation problem [Chou and

a)

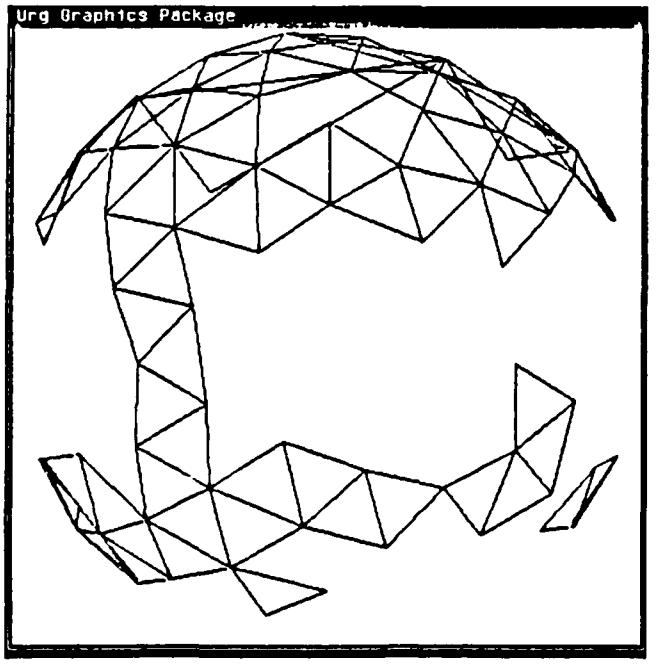


b)





c)



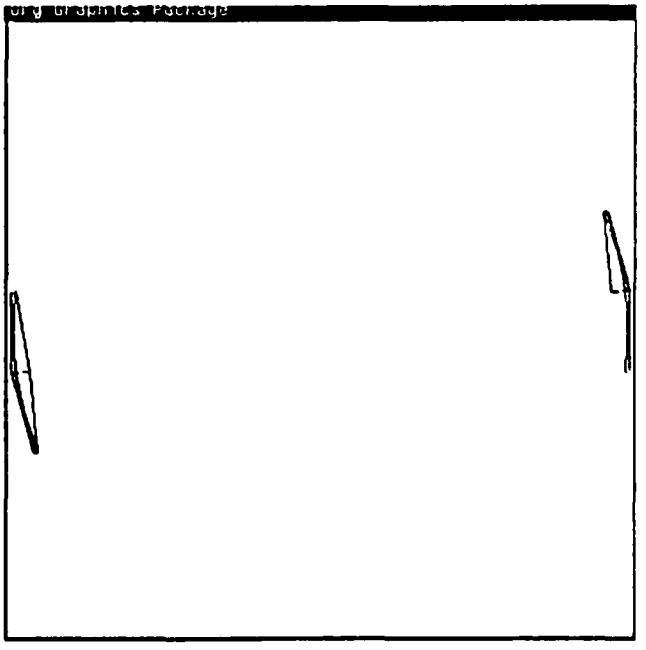


Figure 4: (a) The viewing polyhedron generated by the algorithm. The blob and stick "horse" is at the center.

(b) Two of the views.

(c) The approximate portions of the viewing sphere that yield the views.

Raman 1987; Chou, Brown and Raman 1987]. In this method, a dynamic stability measure based on observations and prior knowledge determines the order in which pieces of the image are examined. The input (Figure 5a) is a camera image of four blocks. The initial likelihoods that a piece of an image belongs to a boundary between segments (e.g., between blocks or between figure and background) is shown in Figure 5b. These likelihoods are generated from the camera image under specific assumptions about the imaging process, noise, and the behavior of segment boundaries [Sher 1987b]. They are combined with prior knowledge modeled as Markov Random Fields, resulting in the a posteriori probability of the possible segmentations. The results of the HCF algorithm (Figure 5d) are compared with the results of a conventional technique based on simulated annealing (Figure 5c). The HCF algorithm produces better results. It requires fewer, and more predictable, computational resources than the annealing technique. (An interactive general-purpose MRF simulator, with extensive graphics and menu-driven control, is shown in the background of the HCF results. This package has been used to compare the performance of various relaxation algorithms based on MRFs.)

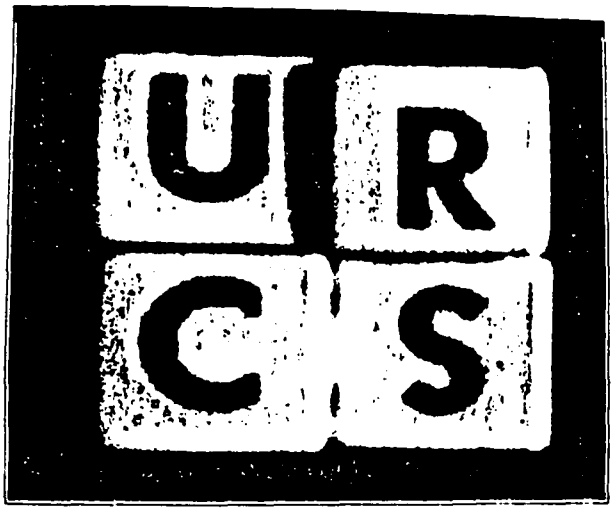
MRFs provide a mechanism for fusing several disparate sources of information [Chou and Brown 1987]. In Figure 6, noise-corrupted orientation and range data images (Figure 6a) provide uncertain and incomplete evidence about the discontinuities or boundaries in the scene. The interactions between image features (edges, boundaries, etc.) are represented as MRFs. The initial MRF encodes only general spatial properties like continuity of edges, and nothing about the location of particular edges. As each source of information is added, its observations are fused with the current MRF to produce a new, a posteriori, MRF that has a stronger bias for specific image features in specific locations. The results of fusing a single source of image data with the initial MRF, followed by running the HCF algorithm to determine the minimal configuration, are shown in Figure 6b. A complete segmentation is achieved (Figure 6c) when all three sources of data are fused with the initial MRF before running the HCF algorithm.

6. Parallel Evidence Combination: by Paul Chou and Bob Potter

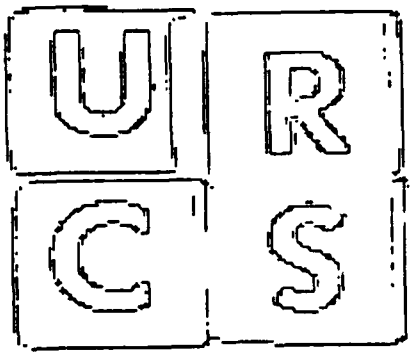
As the algorithm is currently implemented, all of the observational data must be available before interpretation can begin. Potter's current project is to implement the sources of observational data as separate processes on the Butterfly multiprocessor and investigate the performance of algorithm when the data is made available dynamically. In real-world applications, observational data from separate sources will not be produced synchronously, and decisions will be called for even when only partial information is available. If the HCF algorithm performs well under such conditions, it will merit serious consideration in future vision systems.

6.1 Encoding Knowledge in MRFs

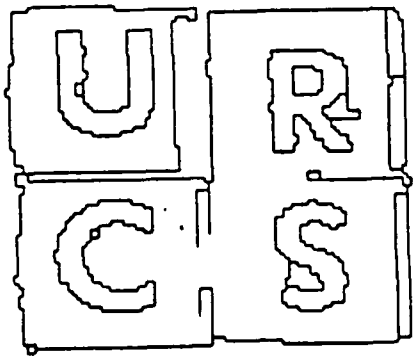
By the Hammersley-Clifford theorem, for an MRF X on a lattice S with a neighborhood system Γ



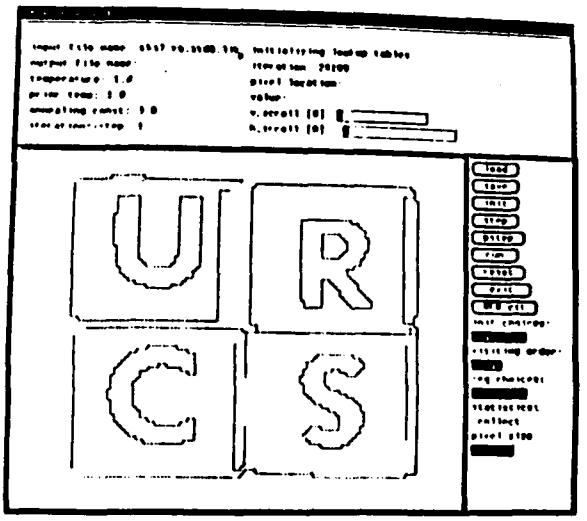
a)



b)



c)



d)

Figure 5: (a) Input image. (b) Initial boundary likelihoods. (c) Output of simulated annealing technique for boundary location. (d) Output of HCF technique for boundary location.

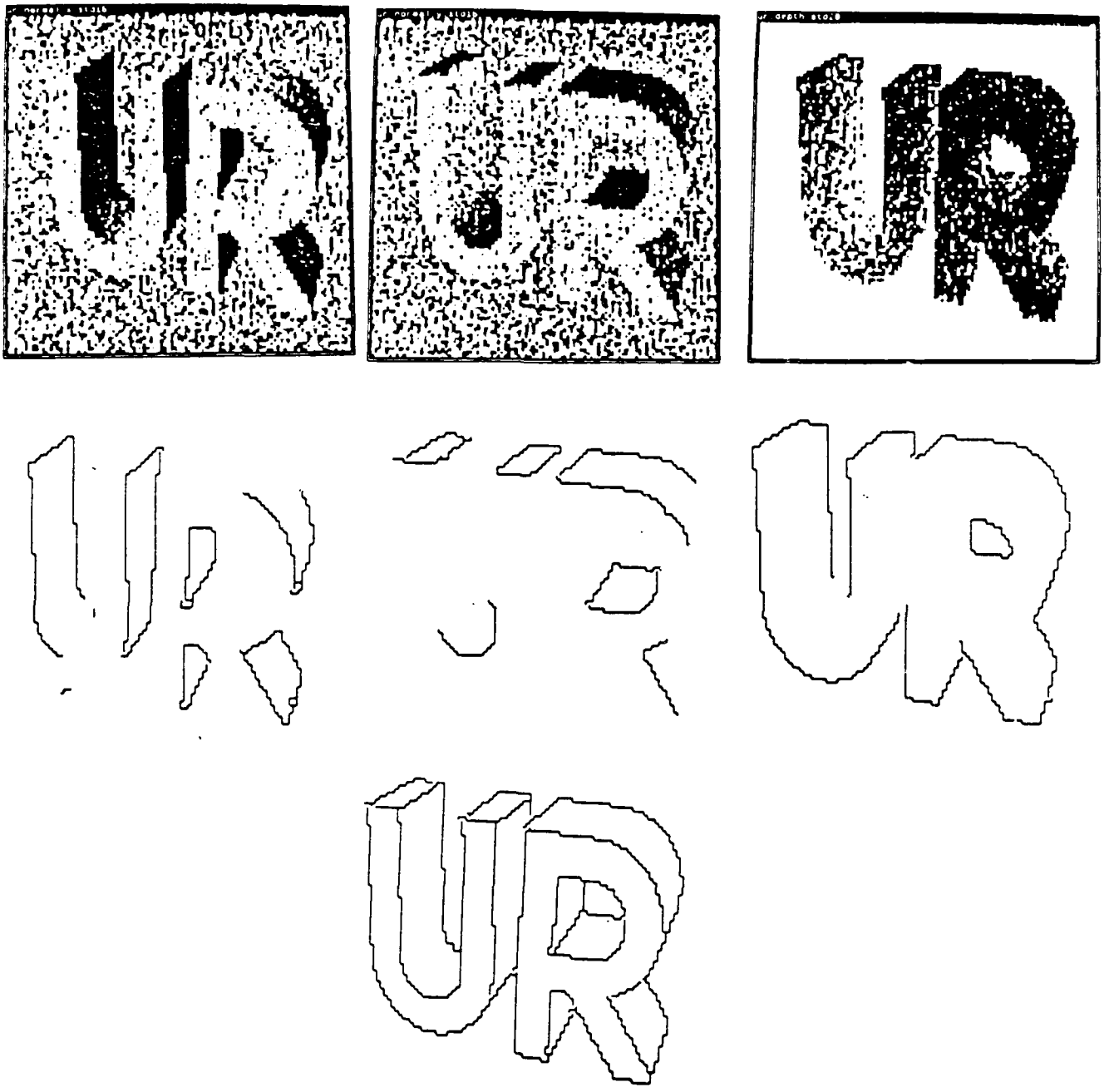


Figure 6: (a) (top row) Noisy synthetic input (depth and orientation). (b) (second row) Individual processing of inputs. (c) (at bottom) Fusion of information from the three sensors.

$$P(X = \omega) = \frac{e^{\frac{-U(\omega)}{T}}}{Z}$$

where ω is a solution from the solution set Ω, Z is a constant normalization factor, and T is the temperature. The energy function $U(\omega)$ is of the form

$$U(\omega) = \sum_{c \in C} V_c(\omega)$$

where C denotes the set cliques (totally connected subgraphs of Γ). V_c represents the potential function of the clique c. The MRF can be characterized by specifying the potential functions V_c for each clique. Chou and Raman define a set of cliques that relate pixel boundaries to their spatial neighbors. Each configuration of adjacent boundaries (such as T join, corner, close parallel or colinear) is assigned a potential that represents a priori knowledge about edge behavior.

The a priori knowledge encoded in the MRF can be combined with image observations with Bayes' rule:

$$P(\omega \mid O) = \frac{P(\omega) P(O \mid \omega)}{\sum_{\omega' \in \Omega} P(\omega') P(O \mid \omega')}$$

where O denotes the image observations. With the assumption that

$$P(O \mid \omega) = \prod_{s \in S} P(O_s \mid \omega_s)$$

we obtain a combined energy function

$$U_O(\omega) = \sum_{c \in C} V_c(\omega) - T \sum_{s \in S} ln\, P(O_s \mid \omega_s)$$

that includes both the a priori and a posteriori knowledge. The probabilities $P(O_S \mid \omega_S)$ are obtained from a boundary detector designed by David Sher [Sher 1987a].

6.2 The Highest Confidence First Algorithm

The HCF algorithm represents decisions for each lattice element (inter-pixel boundary) by one of three labels: EDGE, NON-EDGE, and UNDECIDED. Individual elements make or change their decision in an order based on a dynamic stability measure, with the element that has the most to gain deciding first. The stability measure is simply the energy difference between the element's current decision and its lowest possible one. When one element changes its decision, the stability values of its neighbors are updated according to the MRF formulation given above, and the new least stable element is permitted to change. Eventually every element will reach a stable state. In practice only about 1% of the elements have to change their initial decision.

6.3 Implementation: by Bob Potter

I will be implementing the HCF algorithm on the BBN Butterfly parallel processor using the Structured Message-Passing (SMP) programming environment [LeBlanc et al. 1986]. The a posteriori knowledge sources will be separate processes that can be configured to produce their data in various ways: by scanline, by region, or randomly pixel by pixel. The HCF algorithm process will always attempt to maintain a consistent interpretation of the available data. The knowledge sources themselves are not the focus of this project and initially will simply be dummies that supply pre-computed data. The program will be written so that data from the MaxVideo frame-rate pipelined image processing hardware will be easily accommodated. The output of the program will be to a graphic display on a Sun.

Most of the code for this implementation will come directly from the segmentation simulator for the Sun written by Chou and Raman. My programming contribution will be porting the program to the Butterfly, and writing routines to manage the processes, to communicate pre-computed data from the Sun to the knowledge sources, and to communicate the results back to the Sun. So far I have skeleton routines to handle these Butterfly-specific tasks. My experimental contribution will be the investigation of the performance of the HCF algorithm under several different conditions of data arrival. We are currently planning that data from:

1) random pixels arrive together;
2) scanlines arrive together;
3) scanlines arrive asynchronously;
4) 2D patches of image arrive together;
5) 2D patches of image arrive asynchronously.

We expect the HCF performance to degrade when less than maximal information is available for a decision. The interesting questions are how the performance falls off with loss of spatial coherence and loss of sensor simultaneity in the data.

7. References

Ballard, D.H., "Eye movements and spatial cognition," TR 218, Computer Science Dept., U. Rochester, November 1987.

Ballard, D.H., C.M. Brown, D.J. Coombs, and B.D. Marsh, "Eye movements and computer vision," *1987-88 Computer Science and Engineering Research Review*, Computer Science Dept., U. Rochester, October 1987.

Chou, P.B. and C.M. Brown, "Probabilistic information fusion for multi-model image segmentation," *Proc., Int'l. Joint Conf. on Artificial Intelligence*, Milan, Italy, August 1987.

Chou, P.B., C.M. Brown, and R. Raman, "A confidence-based approach to the labeling problem," *Proc., IEEE Workshop on Computer Vision*, November 1987.

Chou, P.B. and R. Raman, "On relaxation algorithms based on Markov random fields," TR 212, Computer Science Dept., U. Rochester, July 1987.

Coombs, D.J. and B.D. Marsh, "ROVER: A prototype active vision system," TR 219, Computer Science Dept., U. Rochester, August 1987.

Cooper, P.R., "Order and structure in stereo correspondence," TR 216, Computer Science Dept., U. Rochester, June 1987a; submitted, *Int. J. of Computer Vision.*

Cooper, P. R., "Structure recognition by connectionist relaxation," submitted, *1988 Canadian Conference on Artificial Intelligence*, November 1987b.

Cooper, P.R. and S.C. Hollbach, "Parallel recognition of objects comprised of pure structure," *Proc., DARPA Image Understanding Workshop*, pp. 381-391, February 1987.

Cornog, K., "Control of a binocular robot head," Master's Thesis, Massachusetts Inst. of Technology, 1986.

Kitchen and Rosenfeld, "Discrete relaxation for matching relational structures," *IEEE Trans. on Systems, Man and Cybernetics 9*, 12, pp. 869-874, 1979.

LeBlanc, T.J., N.M. Gafter, and T. Ohkami, "SMP: A message-based programming environment for the BBN Butterfly," BPR 8, Computer Science Dept., U. Rochester, 1986.

Sher, D.B., "Advanced likelihood generators for boundary detection," TR 197, Computer Science Dept., U. Rochester, January 1987a.

Sher, D.B., "A probabilistic approach to low-level vision," Ph.D. Thesis and TR 232, Computer Science Dept., U. Rochester, October 1987b.

Swain, M.J., "Decision tree methods for optimal object recognition from a large database," Thesis Proposal, Computer Science Dept., U. Rochester, December 1987.

Watts, N., "Calculating the principal views of a polyhedron," TR 234, Computer Science Dept., U. Rochester, December 1987; submitted, *Int'l. Conf. on Pattern Recognition.*

Evidence Combination Using Likelihood Generators

David Sher
Computer Science Department
The University of Rochester
Rochester, New York 14627

January 1987
TR192

Abstract

Here, I address the problem of combining output of several detectors for the same feature of an image. I show that if the detectors return *likelihoods* I *can robustly* combine their outputs. The combination has the advantages that:

- The confidences of the operators in their own reports are taken into account. Hence if an operator is confident about the situation and the others are not then the reports of the confident operator dominates the decision process.
- *A priori* confidences in the different operators can be taken into account.
- The work to combine 'N' operators is linear in 'N'.

This theory has been applied to the problem of boundary detection. Results from these tests are presented here.

This work would have been impossible without the advice and argumentation of such people as Paul Chou and Mike Swain (who has made suggestions from the beginning) and of course my advisor Chris Brown, and who could forget Jerry Feldman. This work was supported by the Defense Advanced Research Projects Agency U. S. Army Engineering Topographic Labs under grant number DACA76-85-C-0001. Also this work was supported by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, and the Air Force Office of Scientific Research, Bolling AFB, DC 20332, under Contract No. F30602-85-C-0008. This contract supports the Northeast Artificial Intelligence Consortium (NAIC).

1. Introduction

Often in computer vision one has a task to do such as deriving the boundaries of objects in an image or deriving the surface orientation of objects in an image. Often one also has a variety of techniques to do this task. For boundary detection there are a variety of techniques from classical edge detection literature [Ballard82] and the image segmentation literature e.g. [Ohlander79]. For determining surface orientation there are techniques that derive surface orientation from intensities [Horn70] and texture [Ikeuchi80] [Aloimonos85]. These techniques make certain assumptions about the structure of the scene that produced the data. Such techniques are only reliable when their assumptions are met. Here I show that if several algorithms return likelihoods I can derive from them the correct likelihood when at least one of the algorithms' assumptions are met. Thus I derive an algorithm that works well when any of the individual algorithms works well.

The mathematics here were derived independently but are similar to the treatment in [Good50]. and [Good83], using different notation. To understand my results first one must understand the meaning of likelihood.

2. Likelihoods

In this paper I call the assumptions that an algorithm makes about the world a *model*. Most models for computer vision problems describe how configurations in the real world generate observed data. Because imaging projects away information, the models do not explicitly state how to derive the configuration of the real world from the sensor data. As a result, graphics problems are considerably easier than vision problems. Programs can generate realistic images that no program can analyze.

Let O be the observed data, f a feature of the scene whose existence we are trying to determine (like a boundary between two pixels) and M a model. Many computer vision problems can be reduced to finding the probability of the feature given the model and the data, $P(f|O\&M)$. However most models for computer vision instead make it easy to compute $P(O|f\&M)$. I call $P(O|f\&M)$ (inspired by the statistical literature) the *likelihood* of f given observed data O under M. As an example assume f is "the image has a constant intensity before noise". M says that the image has a normally distributed uncorrelated (between pixels) number added to each pixel (the noise). Calculating $P(O|M\&f)$ is straight-forward (a function of the mean and variance of O).

A theorem of probability theory, *Bayes' law*, shows how to derive conditional probabilities for features from likelihoods and prior probabilities. Bayes' law is shown in equation 1.

$$P(f|O\&M) = \frac{P(O|f\&M)P(f|M)}{P(O|f\&M)P(f|M)+P(O|\sim f\&M)P(\sim f|M)} \quad (1)$$

f is the feature for which we have likelihoods. M is the domain model we are using. $P(O|f\&M)$ is the likelihood of f under M and $P(f|M)$ is the probability under M of f

For features that can take on several discrete mutually exclusive labels (rather than just true and false) such as surface orientation (which can be a pair of angles to the nearest degree or "not applicable" (at boundaries)) a more complex form of Bayes' law shown in equation 2 yields conditional probabilities from likelihoods and priors.

$$P(l|O\&M) = \frac{P(O|l\&M)P(l|M)}{\sum_{l' \in L(f)} P(O|l'\&M)P(l'|M)} \quad (2)$$

l is a label for feature f and $L(f)$ is the set of all possible labels for feature f.

Another important use for explicit likelihoods is for use in Markov random fields. Markov random fields describe complex priors that can capture important information. Several people have applied Markov random fields to vision problems [Geman84]. Likelihoods can be used in a Markov random field formulation to derive estimates of boundary positions [Marroquin85b] [Chou87]. In [Sher86] and [Sher87] I discuss algorithms for determining likelihoods of boundaries.

Let us call an algorithm that generates likelihoods a *likelihood generator*. Different models lead to different likelihood generators. The difference between two likelihood generators' models can be a single constant (such as the assumed standard deviation of the noise) or the two likelihood generators' models may not resemble each other in the slightest

Consider likelihood generators L_1 and L_2 with models M_1 and M_2 and assume they both determine probability distributions for the same feature. L_1 can be considered to return the likelihood of a label l for feature f given observed data O and the domain model M_1. Thus L_1 calculates $P(O|f=l\&M_1)$. Also L_2 calculates $P(O|f=l\&M_2)$. A useful combination of L_1 and L_2 is the likelihood detector that returns the likelihoods for the case where M_1 or M_2 is true Also the prior confidences one has in M_1 and M_2 should be taken into account.

This paper studies deriving $P(O|f=1\&(M_1 \vee M_2))$. Note that if I can derive rules for combining likelihoods for two different models then by applying the combination rules N times, N likelihoods are combined. Thus all that is needed is combination rules for two models.

3. Combining Likelihoods From Different Models

To combine likelihoods derived under M_1 and M_2 an examination of the structure and interaction of the two models is necessary. M_1 and M_2 must have the same definition for the feature being detected. If the feature is defined differently for M_1 and M_2 then M_1 and M_2 are about different events, and the likelihoods can not be combined with the techniques developed in this section.

Thus the likelihood generated by an occlusion boundary detector can not be combined with the likelihood generated by a detector for boundaries within the image of an object (such as corners internal to the image). A detector of the likelihood of heads on a coin flip can not be combined with a detector of the likelihood of rain outside using this theory. (However easy it may be using standard probability theory.)

If the labeling of a feature f implies a labeling for another feature g then in theory one can combine a f detector with a g detector by using the g detector that is implied by the f detector. As an example a region grower could be combined with a boundary detector since the position of the regions implies the positions of the boundaries.

3.1. Combining Two Likelihoods

The formula for combining the likelihoods generated under M_1 and M_2 requires prior knowledge. Necessary are the prior probabilities $P(M_1)$ and $P(M_2)$ that the domain models M_1 and M_2 are correct as well as $P(M_1\&M_2)$. Often $P(M_1\&M_2) = 0$. When this occurs the two models contradict each other. I call two such models *disjoint* because both can not describe the situation

simultaneously. If M_1 is a model with noise of standard deviation $4 \pm \varepsilon$ and M_2 is a model with noise of standard deviation $8 \pm \varepsilon$ then their assumptions contradict and $P(M_1 \& M_2) = 0$.

Prior probabilities for the feature labels under each model ($P(f=l|M_1)$ and $P(f=l|M_2)$) are necessary. If $P(M_1 \& M_2) \neq 0$ then the prior probability of the feature label under the conjunction of M_1 and M_2 ($P(f=l|M_1 \& M_2)$) and the output of a likelihood generator for the conjunction of the two models ($P(O|f=l \& (M_1 \& M_2))$) are needed. If I have this prior information I can derive $P(O|f=l \& (M_1 \vee M_2))$.

If I were to combine another model, M_3, with this combination I need the priors $P(M_3)$, $P(f|M_3)$, $P(M_3 \& (M_1 \vee M_2))$ and $P(f|M_3 \& (M_1 \vee M_2))$. To add on another model I need another 4 priors. Thus the number of prior probabilities to combine n models is linear in n.

Thus all that is left is to derive the combination rule for likelihood generators given this prior information. The derivation starts by applying the definition of conditional probability in equation 3

$$P(O|f=l \& (M_1 \vee M_2)) = \frac{P(O \& f=l \& (M_1 \vee M_2))}{P(f=l \& (M_1 \vee M_2))} \tag{3}$$

The formula for probability of a disjunction is applied to the numerator and denominator in equation 4.

$$P(O|f=l \& (M_1 \vee M_2)) = \frac{P(O \& f=l \& M_1) + P(O \& f=l \& M_2) - P(O \& f=l \& M_1 \& M_2)}{P(f=l \& M_1) + P(f=l \& M_2) - P(f=l \& M_1 \& M_2)} \tag{4}$$

In equation 5 the definition of conditional probability is applied again to the terms of the numerator and the denominator.

$$P(O|f=l \& (M_1 \vee M_2)) = \frac{\left[\begin{array}{c} P(O|f=l \& M_1)P(f=l|M_1)P(M_1) \\ + \\ P(O|f=l \& M_2)P(f=l|M_2)P(M_2) \\ - \\ P(O|f=l \& M_1 \& M_2)P(f=l|M_1 \& M_1)P(M_1 \& M_2) \end{array}\right]}{P(f=l|M_1)P(M_1) + P(f=l|M_2)P(M_2) - P(f=l|M_1 \& M_2)P(M_1 \& M_2)} \tag{5}$$

Different assumptions allow different simplifications to be applied to the rule in equation 5. If the two models are disjoint equation 5 reduces to equation 6.

$$P(O|f=l \& (M_1 \vee M_2)) = \frac{\left[\begin{array}{c} P(O|f=l \& M_1)P(f=l|M_1)P(M_1) \\ + \\ P(O|f=l \& M_2)P(f=l|M_2)P(M_2) \end{array}\right]}{P(f=l|M_1)P(M_1) + P(f=l|M_2)P(M_2)} \tag{6}$$

Another assumption that simplifies things considerably is the assumption that prior probabilities for all feature labelings in all the models and combinations thereof are the same. I call this assumption *constancy of priors*. When constancy of priors is assumed $P(f=l|M_1) = P(f=l|M_2) = P(f=l|M_1 \& M_2)$. Making this assumption reduces the number of priors that need to be determined. Since determining prior probabilities from a model is sometimes a difficult task the constancy of priors is a useful simplification. With constancy of priors equation 5 reduces to equation 7.

$$P(O|f=l\&(M_1 \vee M_2)) = \frac{\left[\begin{array}{c} P(O|f=l\&M_1)P(M_1) \\ + \\ P(O|f=l\&M_2)P(M_2) \\ - \\ P(O|f=l\&M_1\&M_2)P(M_1\&M_2) \end{array}\right]}{P(M_1)+P(M_2)-P(M_1\&M_2)} \quad (7)$$

Equation 6 with constancy of priors reduces to equation 8.

$$P(O|f=l\&(M_1 \vee M_2)) = \frac{\left[\begin{array}{c} P(O|f=l\&M_1)P(M_1) \\ + \\ P(O|f=l\&M_2)P(M_2) \end{array}\right]}{P(M_1)+P(M_2)} \quad (8)$$

Thus equation 8 describes the likelihood combination rule with disjoint models and constancy of priors.

3.2. Understanding the Likelihood Combination Rule

The easiest incarnation of the likelihood combination rule to understand is the rule for combining likelihoods from disjoint models given constancy of priors across models (equation 8). Here the combined likelihood is the weighted average of the likelihoods from the individual models weighted by the probabilities of the models applying. (The combined likelihood is the likelihood given the disjunction of the models).

If models M_1 and M_2 are considered equally probable and the likelihoods returned by M_1's detector are considerably larger than those of M_2's detector then the probabilities determined from the combination of M_1 and M_2 are close to those determined from M_1. Thus a model with large likelihoods determines the probabilities. To illustrate this principle consider an example.

Assume that a coin has been flipped $n+1$ times. The results of flipping it has been reported for the first n times. The task is to determine the probability of heads having been the result of the $n+1^{st}$ flip. Consider the results of each coin flip independent. Let M_1 be the coin being fair so that the probability of heads and tails is equal. Let M_2 be that the coin is biased with the probability of heads is π and tails $1-\pi$ with π being a random choice with equal probability between p and $1-p$. Hence the coin is biased towards heads or tails with equal probability but the bias is consistent between coin tosses. The probability of heads remains the same for all coin tosses in both models. M_1 and M_2 are disjoint (the coin is either fair or it isn't but not both) and the prior probability of a flip being heads or tail is the same for both, .5.

Under M_1 the probability of each of the possible flips of $n+1$ coins is 2^{-n-1}. Under M_2 the probability of $n+1$ flips of coins with h heads and $t=n+1-h$ tails is:

$$\frac{1}{2}p^h(1-p)^t + \frac{1}{2}p^t(1-p)^h$$

Let $n=2$ and $p=.9$. Assume the first two flips are both heads. Let H be "the third flip was heads" and T be "the third flip was tails." The likelihood of H given the observed data is the probability of all 3 flips being heads divided by the probability of the third flip being heads. The likelihood of T given the observed data is the probability of the first 2 being heads and the 3rd tails divided by the probability of the third flip being tails.

Under M_1 the probability of all 3 flips being heads is 0.125 and the probability of a flip being heads is 0.5 thus the likelihood of H is 0.25. The likelihood of T is 0.25 by the same reasoning

Applying Bayes' law to get the probability of H under M_1 one derives a probability of .5 .

Under M_2 the probability of all 3 flips being heads is 0.365 and the probability of a flip being heads is 0.5. Thus the likelihood of H is 0.73. Under M_2 the probability of the first two being heads and the third being tails is 0.045 and the probability of a flip being tails is 0.5. Thus the likelihood of T is 0.09. Applying Bayes' law under M_2 a probability of H being 0.89 is derived.

If M_1 and M_2 are considered equally probable then the combination of the likelihoods from the two models is the average of the two likelihoods. Thus the likelihood of H for this combination is 0.49 and the likelihood of T is 0.17 (likelihoods don't have to sum to 1). Bayes' law combines these probabilities to get 0.74 for the 3rd flip to be heads.

The table in figure 1 describes combining various M_2's with different values of p with M_1 for the different combinations with $n=4$

Observed	Combined with M_1	Likelihood of H		Likelihood of T		Probability of H	
Coin Flips	or just M_2	$p=.6$	$p=.9$	$p=.6$	$p=.9$	$p=.6$	$p=.9$
HHHH	Just M_2	0.088	0.5905	0.0672	0.0657	0.567	0.8999
	Combined	0.07525	0.3265	0.06485	0.0641	0.537	0.8359
HHHT	Just M_2	0.0672	0.0657	0.0576	0.0081	0.5385	0.8902
	Combined	0.06485	0.0641	0.06005	0.0353	0.5192	0.6449
HHTT	Just M_2	0.0576	0.0081	0.0575	0.0081	0.5	0.5
	Combined	0.06005	0.0353	0.06	0.0353	0.5	0.5
HTTT	Just M_2	0.0576	0.0081	0.0672	0.0657	0.4615	0.1098
	Combined	0.06005	0.0353	0.06485	0.0641	0.4808	0.3551
TTTT	Just M_2	0.0672	0.0657	0.088	0.5905	0.433	0.1001
	Combined	0.06485	0.0641	0.07525	0.3265	0.4629	0.1641

Figure 1: Result of likelihood combination Rule

Look at the probabilities with $p=.9$ and the observed data is HHHH. For this case the observed data fits M_2 much better than M_1 and the probability from combining M_1 and M_2 is close to the probability resulting from using just M_2, .9. If we had a longer run of heads the probability of future heads would approach exactly M_2's prediction, .9. On the other hand if we had a long run of equal numbers of heads and tails the probability of future heads would quickly approach the prediction of M_1, .5. When the observed data is HHHT the observed data fits M_1 about as well as M_2 and the resulting probability is near the average of .5 predicted by M_1 and 0.8902 predicted by M_2. Thus when the observed data is a good fit for a particular model (like M_2) the probabilities predicted by the combination is close to the probabilities predicted by the fitted model. If two models fit about equally then the result is an average of the probabilities [1].

4. When No Model Applies

Given a set of likelihood generators and their models, using the evidence combination described in section 3 we can get the likelihood for the feature labelings given that at least one model applies. Thus if we have likelihoods of a boundary given models with the noise standard

[1]However the feature that the decision theory predicts is not the average of the features predicted under the two different models in general

deviations near to 4, 8 and 16 in them we can derive the likelihood of a given the noise standard deviation is near to 4 or 8 or 16 (no matter which). Thus we can derive the probability distribution over feature labelings given that at least one of our models applies. However what we are trying to derive is the *physical probability distribution* over the feature labelings. This is the probability distribution over feature labels given the observed data (estimated by the long run frequencies over the feature labels given the observed data). The problem is that there may be a case where none of the models assumptions is true. In the Venn diagram of figure 2 each set represents the set of situations where a model's assumptions are true. The area marked NO MODEL is the set of situations where all the models fail.

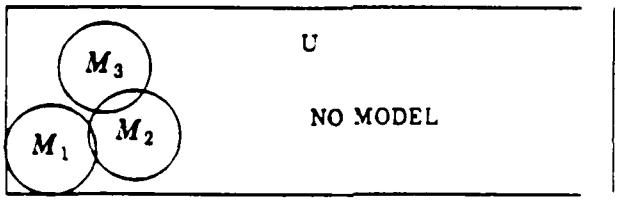


Figure 2: Venn Diagram of Models

What should the likelihood of a feature label be if no model applies? To answer this question I examine the companion question of what should the probability of a feature label be if no model applies. Assume a prior probability for the label is available. If a posterior probability is different from a prior probability for the feature then information has been added to get the posterior. (Only information can justify changing from the prior.) Since having no model means intuitively having no information then the posterior should be the same as the prior. If and only if the likelihoods of all feature labels are equal, the posterior probability is the same as the prior. Hence the likelihoods of the feature labels should be equal for any particular piece of observed data. In this section I assume a prior proability distribution is a available over feature labels. If no such distribution is available an uninformative prior can be constructed [Frieden85].

To constrain the problem further, consider whether any piece of observed data should be more probable than any other when no model applies. It seems unreasonable that one could conclude that some observations are more probable than others without any model of how those observations were produced. Hence all the likelihoods should be equal This constraint is sufficient to determine the likelihoods when no model applies. I think that this solution minimizes cross entropy with the prior (since it returns the prior) [Johnson85].

To derive the physical probability distribution over feature labels, the "no model" likelihoods should be combined with the likelihoods derived for the models. The probability of each of the models and their combinations must have been available to use the combination rules from section 3. Hence the probability that one or more of the models applies is known. The probability of no model is 1 minus that probability. The conjunction of some model applying and no model applying has 0 probability. Hence combination rule 6 can be applied to derive the likelihoods under any conditions from the likelihoods for any model applying.

As example consider the problem of seeing HHHH and trying to derive the probability of a fifth head given the equally likely choices that the coin is fair or is biased to .9 (biased either for heads or tails with equal probability). The combined likelihood of H is 0.3265 (from figure 1). The combined likelihood of T is 0.0641. As an example, assume that the probabilities that the assumptions of M_1 were true was 0.4 and similar for M_2. Then 0.4 of the time we feel the coin is

fair, 0.4 of the time we feel it has been biased by 0.9, and 0.2 of the time we have no model about what happened. The likelihood of HHHH under "NO MODEL" is .0625 regardless of H or T (Since the likelihood of all 4 coin flip events are equal and must sum to 1). Combining the "NO MODEL" likelihoods with likelihoods of 0.2737 for H and 0.06378 for T (see figure 1), the probability of H from applying Bayes' law to these likelihoods is 0.811. This probability is somewhat nearer to .5 than the probability of 0.8359 derived without taking the possibility of all the models failing into account.

Taking the possibility of all models failing lends certain good properties to the system. Probabilities of 0 or 1 become impossible without priors of 0 or 1. Thus the system is denied total certainty. Numbers near 0 or 1 cause singularities in the equations under finite precision arithmetic. Total certainty represents a willingness to ignore all further evidence. I find that property undesirable in a system. Denying the system total certainty also results in the property that the system must have all probability distribution over feature labels between ϵ and $1-\epsilon$ for an ϵ proportional to the probability that no model applies. Thus there is a limit to how certain our system is about any feature labeling in our uncertain world.

5. Results

I have applied this evidence combination to the boundary detection likelihood generators described in [Sher87]. Here I prove my claims that the evidence combination theory allows me to take a set of algorithms that are effective but not robust and derive an algorithm that is robust. The output of such an algorithm is almost as good as the best of its constituents (the algorithms that are combined).

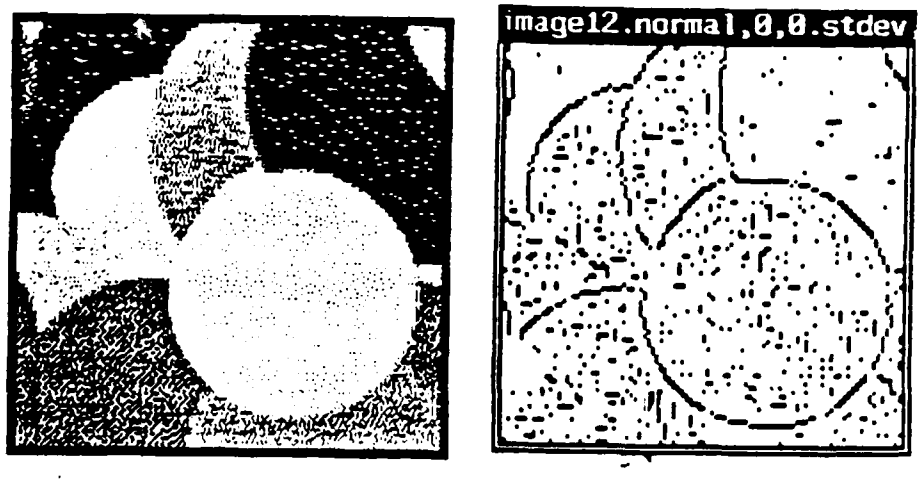
5.1. Artificial Images

Artificial images were used to test the algorithms described in section 3 quantitatively. I used as a source of likelihoods the routines described in [Sher87]. Because the positions of the boundaries in an artificial image are known one can accurately measure false positive and negative rates for different operators. Also one can construct artificial images to precise specifications. The artificial images I use is an image composed of overlapping circles with constant intensity and aliasing at the boundaries shown in figure 3.

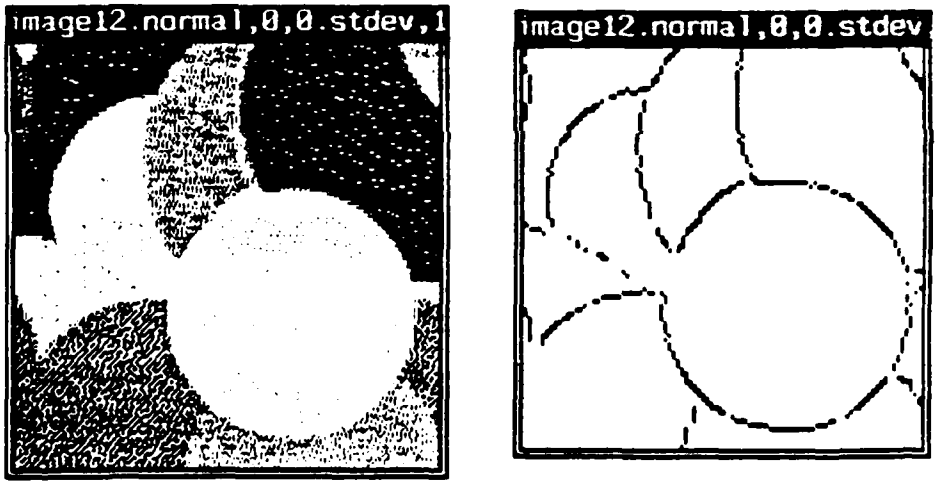
Figure 3: Artificial Test Image

The intensities of the circles were selected from a uniform distribution from 0 to 254. To the circles were added normally distributed uncorrelated noise with standard deviations 4, 8, 12, 16, 20, and 32. The software to generate images of this form was built by Myra Van Inwegen working under my direction. This software will be described in an upcoming technical report.

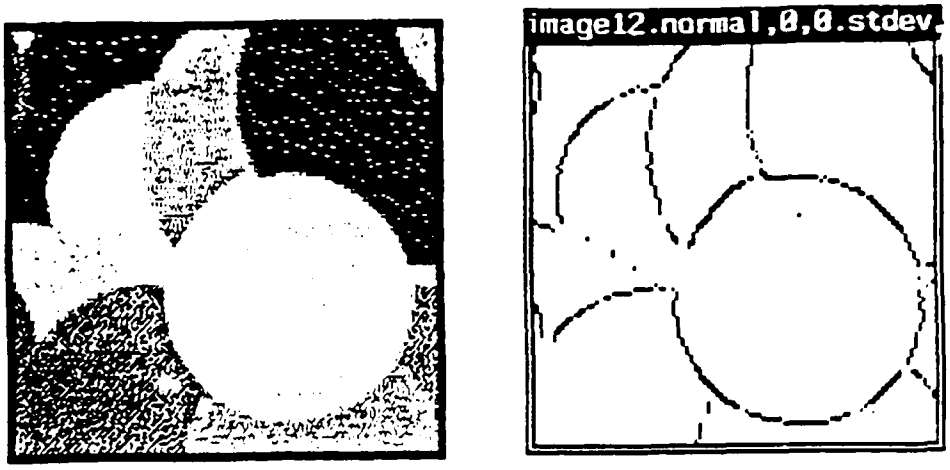
In figure 4 I show the result of applying the detector tuned to standard deviation 4 noise to the artificial image with standard deviation 12 noise added to it. In figure 5 I show the result of applying the detector tuned to standard deviation 12 noise to an image with standard deviation 12 noise added to it. In figure 6 I show the result of applying the combination of the detectors tuned to 4, 8, 12, and 16 standard deviation noise. The combination rule was that for disjoint models with the same priors. The 4 models were combined with equal probability. These operator outputs are thresholded at 0.5 probability with black indicating an edge and white indicating no edge.



a: Image with $\sigma = 12$ noise b: Output of $\sigma = 4$ detector
Figure 4: $\sigma = 4$ detector applied to 3 image with $\sigma = 12$ noise



a: Image with $\sigma = 12$ noise b: Output of $\sigma = 12$ detector
Figure 5: $\sigma = 12$ detector applied to 3 image with $\sigma = 12$ noise



a: Image with $\sigma = 12$ noise b: Output of combined detector
Figure 6: Combined detector applied to 3 image with $\sigma = 12$ noise

Note that the result of using the combined operator is similar to that of the operator tuned to the correct noise level. Most of the false boundaries found by the $\sigma = 4$ operator are ignored by the combined operator.

Using this artificial image I have acquired statistics about the behavior of the combined detector vs the tuned ones under varying levels of noise. Figure 7 shows the false positive rate for the detector tuned to standard deviation 4 noise as the noise in the image increases[2]. Figure 8 shows the false positives for the standard deviation 12 operator. Figure 9 shows the false positive rate for the operator tuned to the current standard deviation of the noise. Figure 10 shows the false positive rate of the combined operator. Figure 11 shows the superposition of the 4 previous graphs.

[2]The operators are thresholded at 0.5 probability to make the decisions about where the boundaries are.

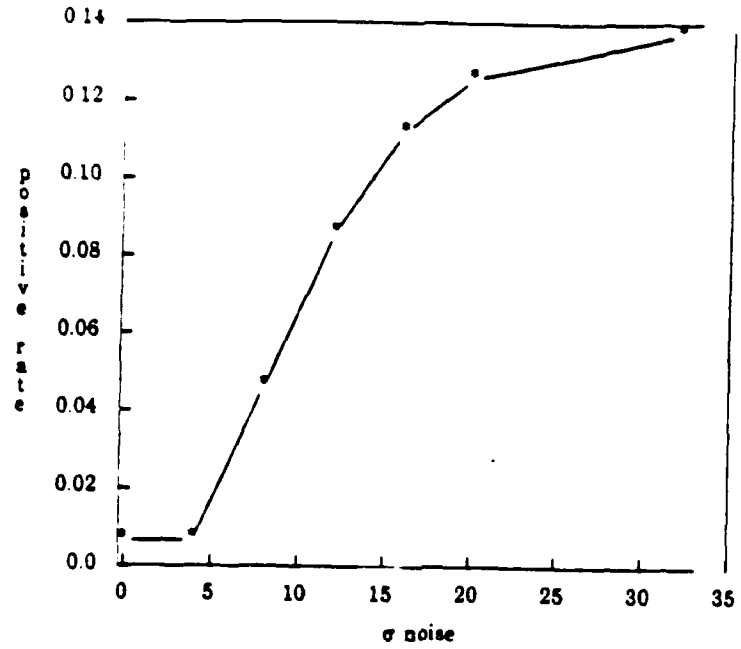


Figure 7: False positives vs noise σ for operator tuned to $\sigma = 4$ noise

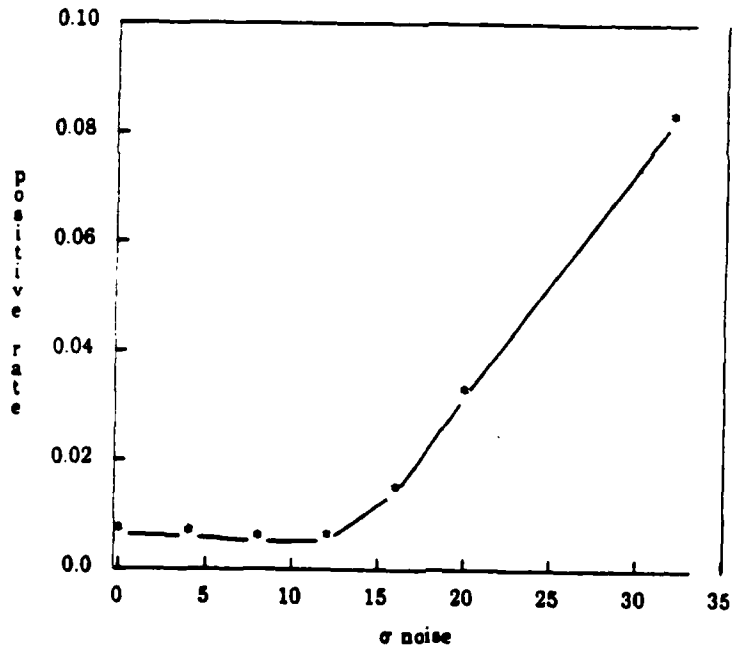


Figure 8: False positives vs noise σ for operator tuned to $\sigma = 12$ noise

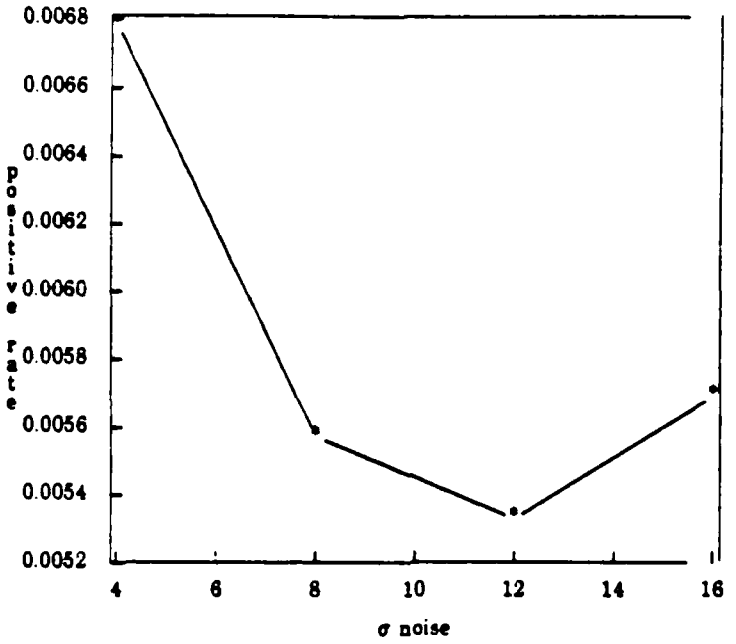


Figure 9: False positives vs noise σ for operator tuned to the noise

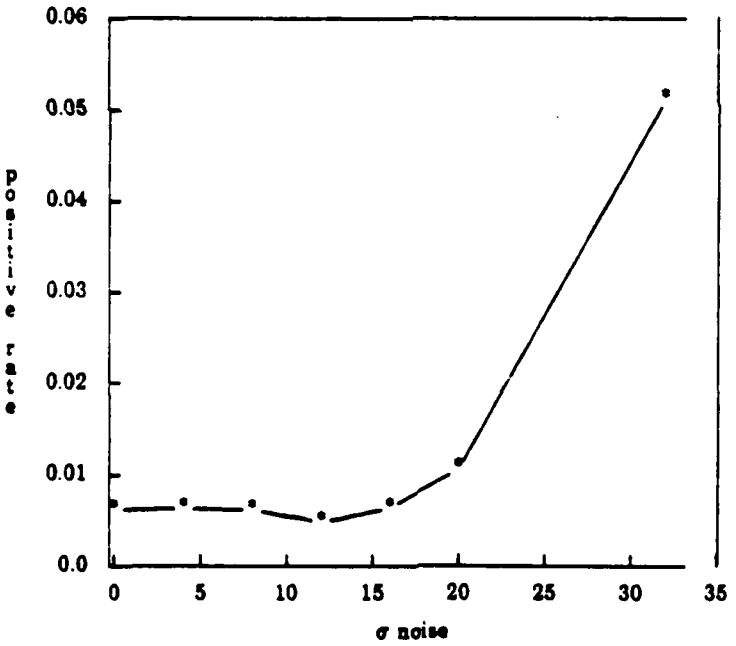
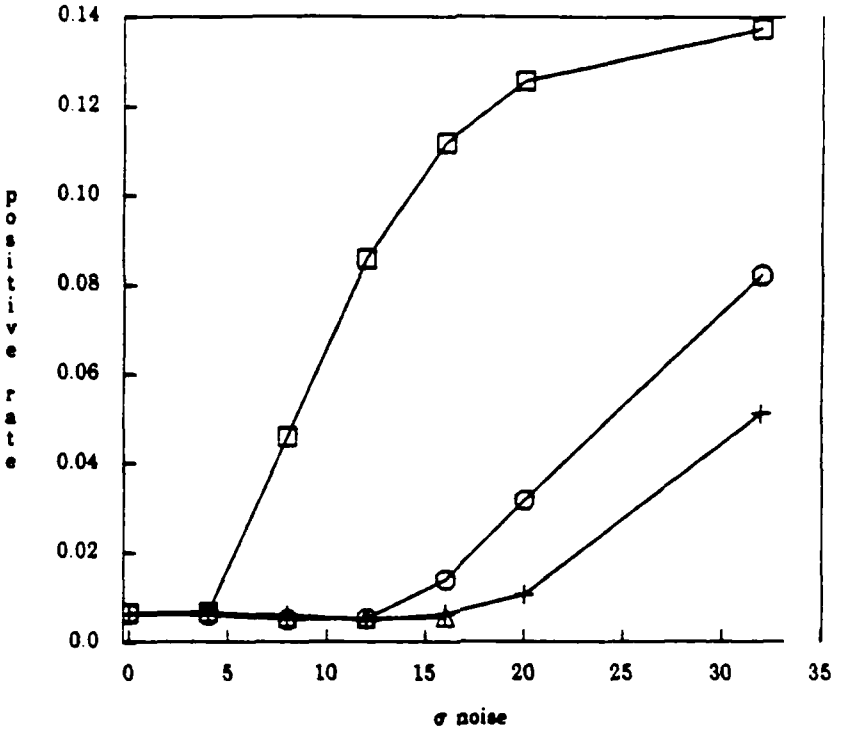


Figure 10: False positives vs noise σ for combined operator



square: $\sigma = 4$ operator circle: $\sigma = 12$ operator
triangle: tuned operator cross: combined operator

Figure 11: False positives vs noise σ for all operators

Note that the combined operator has a false positive rate that is as least as good as that of the tuned operators.

I can also count false negatives. When I counted false negatives I ignored missed boundaries that had an boundary reported one pixel off normal to the boundary (because such an error is a matter of discretization rather than of a more fundamental sort). See figure 12 for an example of a 1 pixel off error.

MISS GOOD

MISS is recorded as a false negative
GOOD is recorded as a true positive

Figure 12: Example of one pixel off error

Figure 13 shows the false negative rate for the detector tuned to standard deviation 4 noise as the noise in the image increases. Figure 14 shows the false negatives for the standard deviation 12 operator. Figure 15 shows the false negative rate for the operator tuned to the current standard deviation of the noise. Figure 16 shows the false negative rate of the combined operator. Figure 17 shows the superposition of the 4 previous graphs.

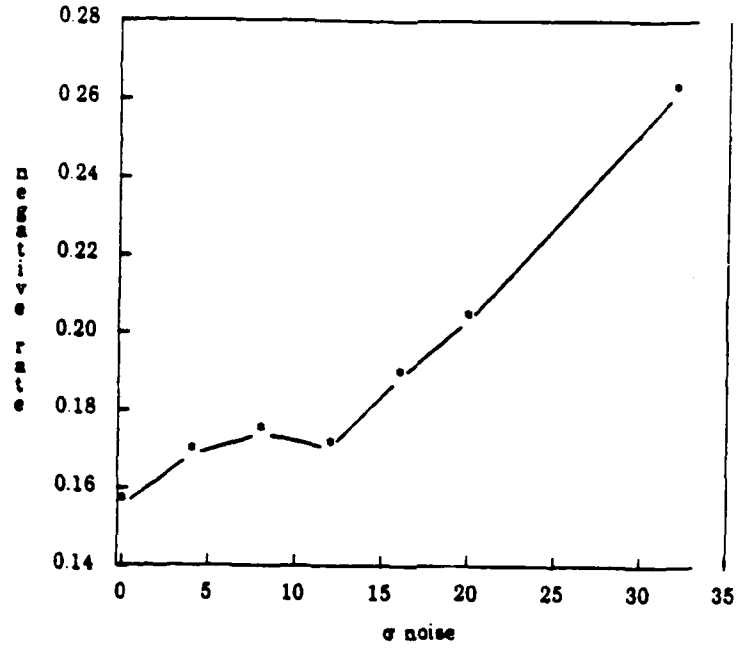


Figure 13: False negative rate for $\sigma = 4$ operator

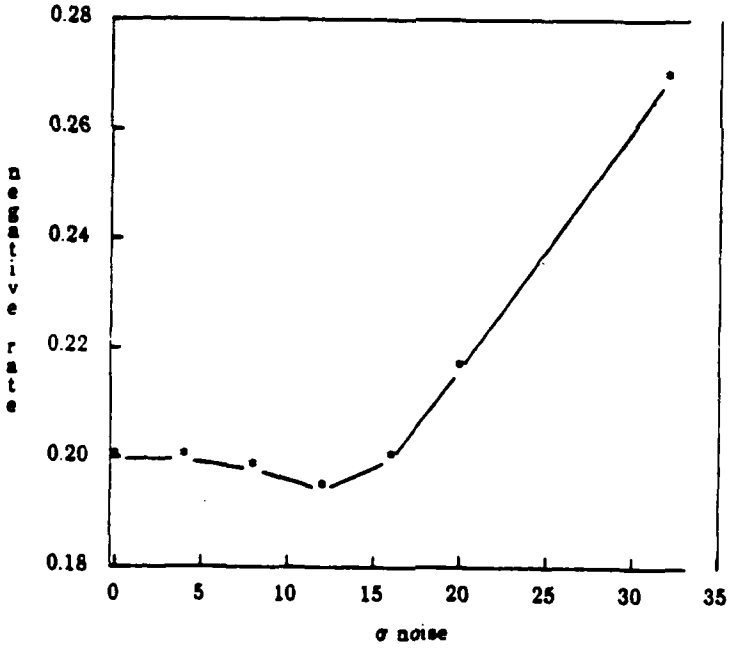


Figure 14: False negative rate for $\sigma = 12$ operator

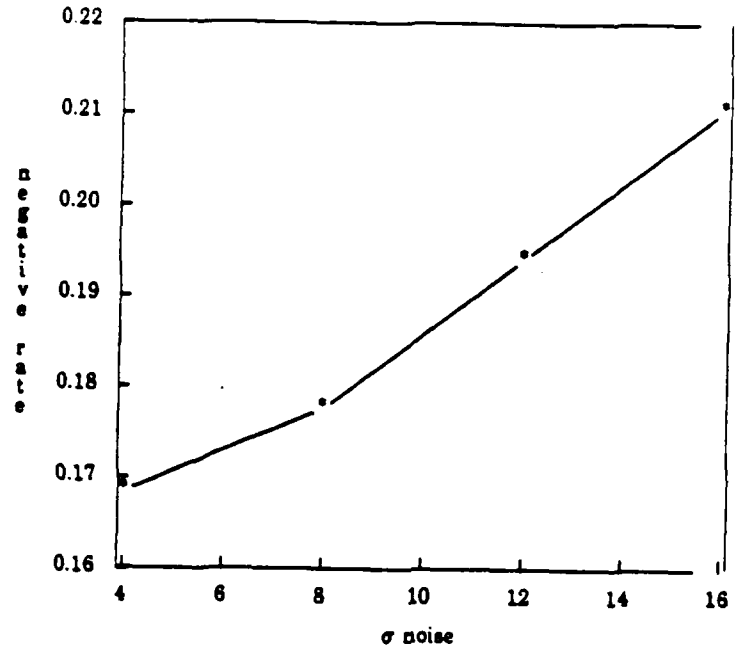


Figure 15: False negative rate for tuned operator

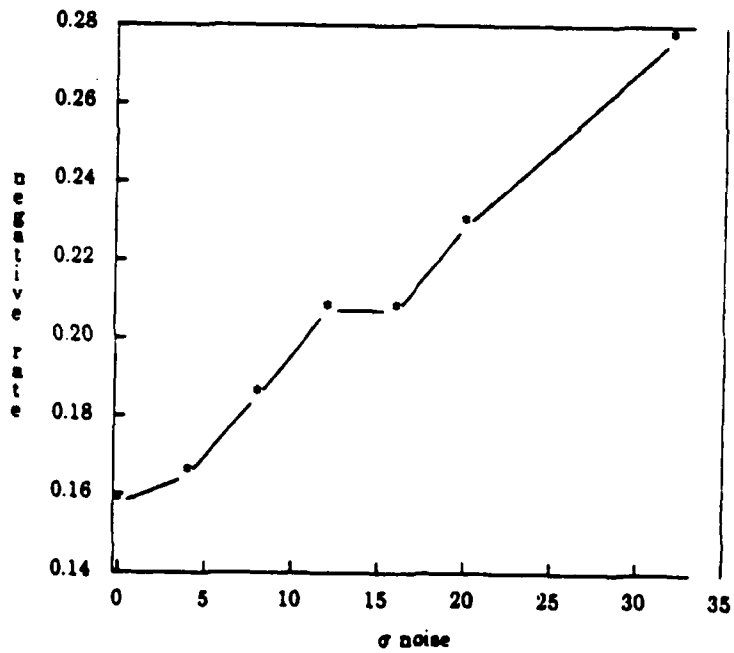
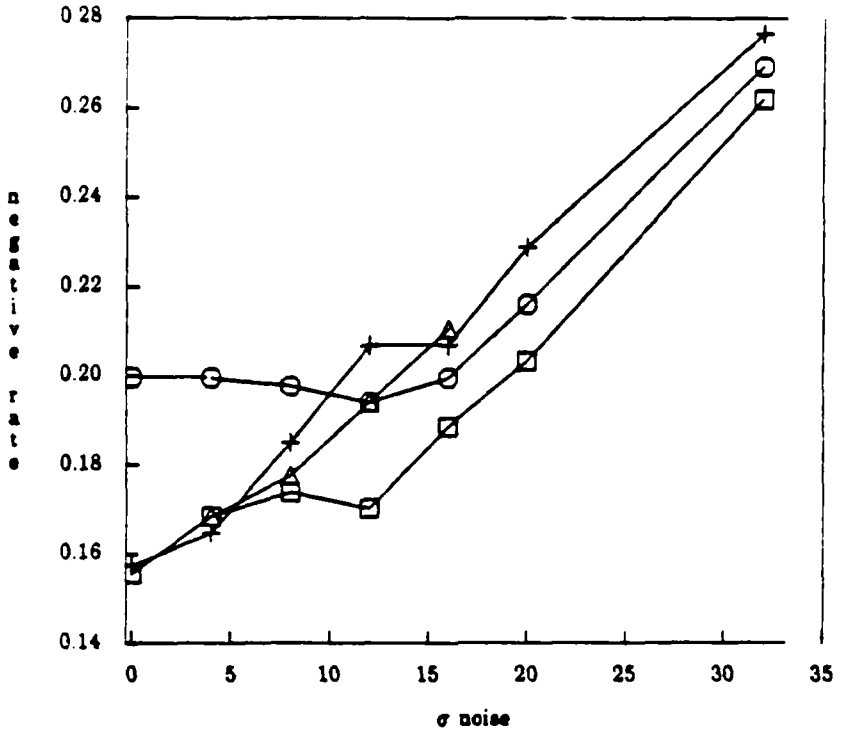


Figure 16: False negative rate for combined operator



square: $\sigma = 4$ operator circle: $\sigma = 12$ operator
triangle: tuned operator cross: combined operator

Figure 17: False negative rate for all operators

Here the combined operator is not always as good as the tuned operators. One must ask if this tendency of the combined operator to miss edges offsets its better performance for false positives. The next series of figures charts the total error rate for the same cases. Figure 18 shows the error rate for the detector tuned to standard deviation 4 noise as the noise in the image increases. Figure 19 shows the error rate for the standard deviation 12 operator. Figure 20 shows the error rate for the operator tuned to the current standard deviation of the noise. Figure 21 shows the error rate of the combined operator. Figure 22 shows the superposition of the 4 previous graphs.

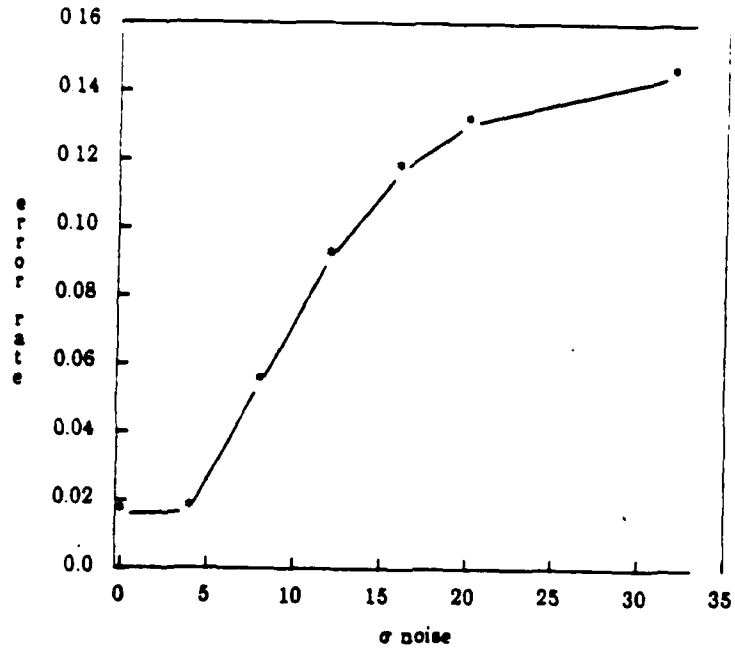


Figure 18: Total errors by the $\sigma = 4$ detector

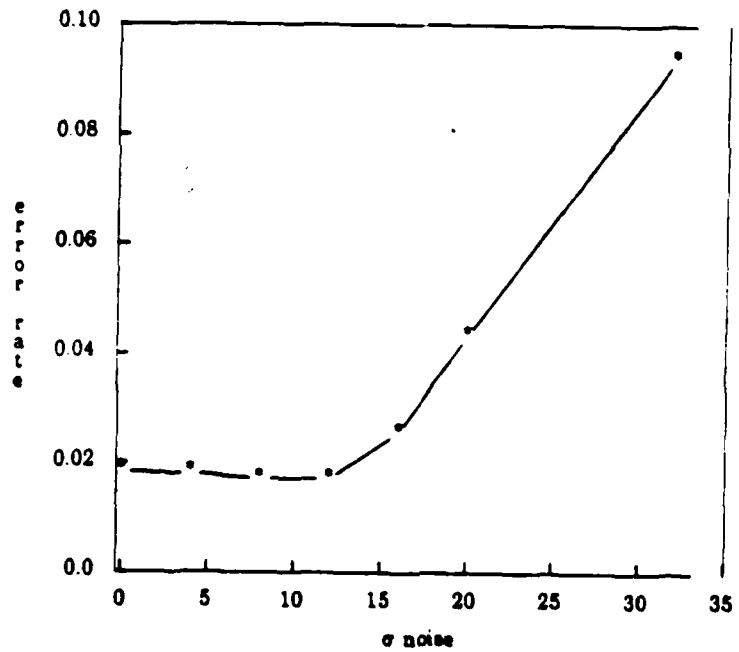


Figure 19: Total errors by the $\sigma = 12$ detector

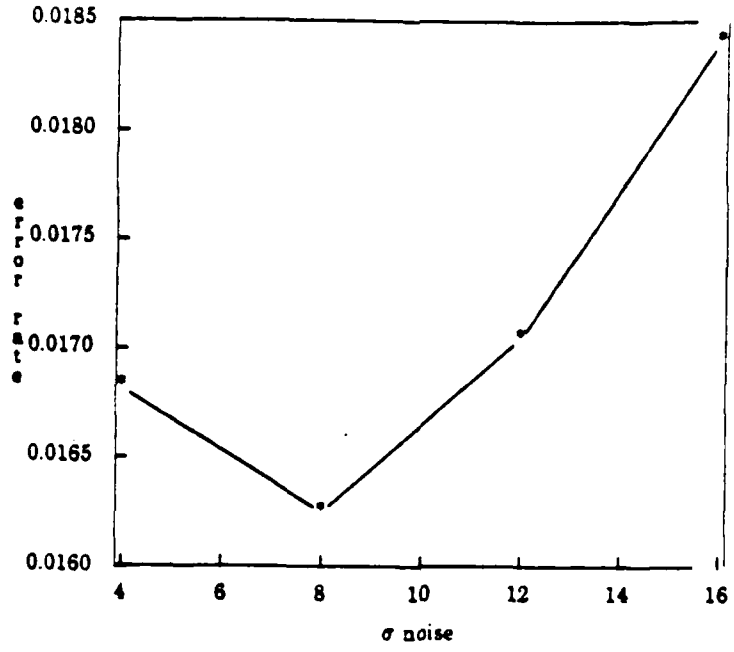


Figure 20: Total errors by the tuned detector

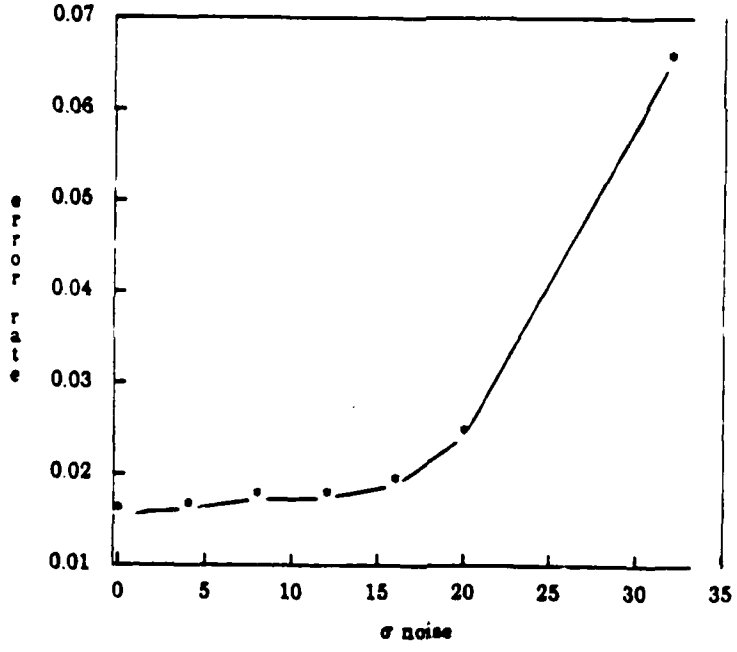
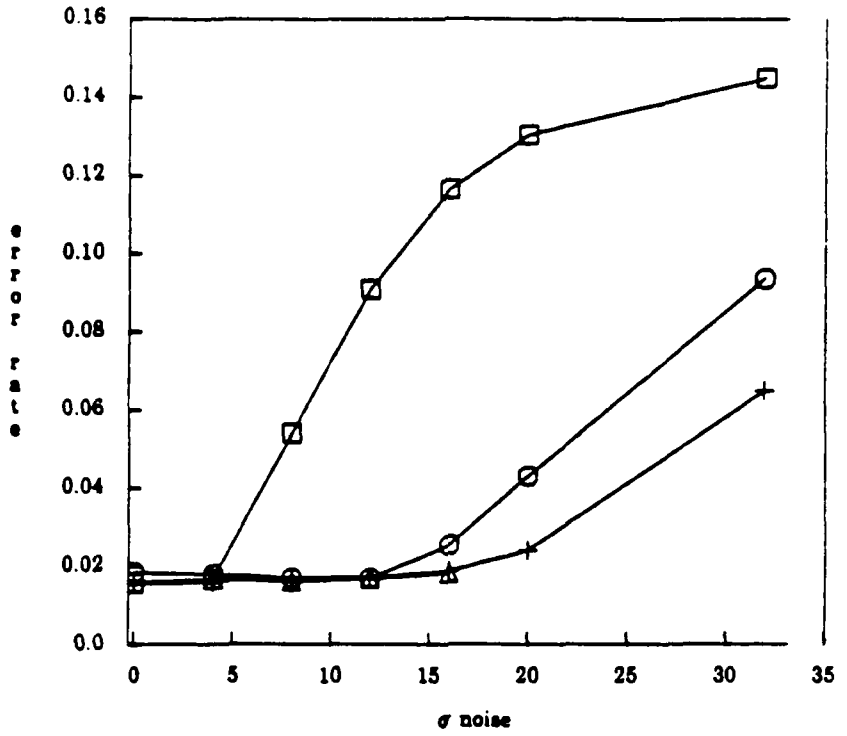


Figure 21: Total errors by the combined detector



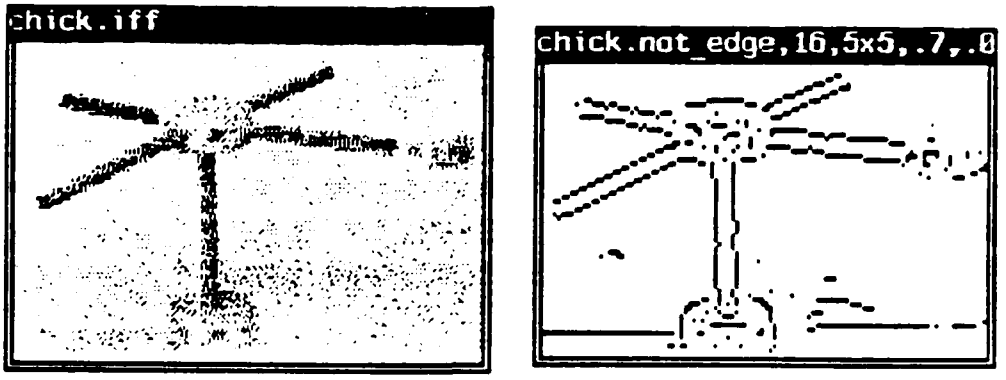
square: $\sigma = 4$ operator circle: $\sigma = 12$ operator
triangle: tuned operator cross: combined operator

Figure 22: Total errors by the all detectors

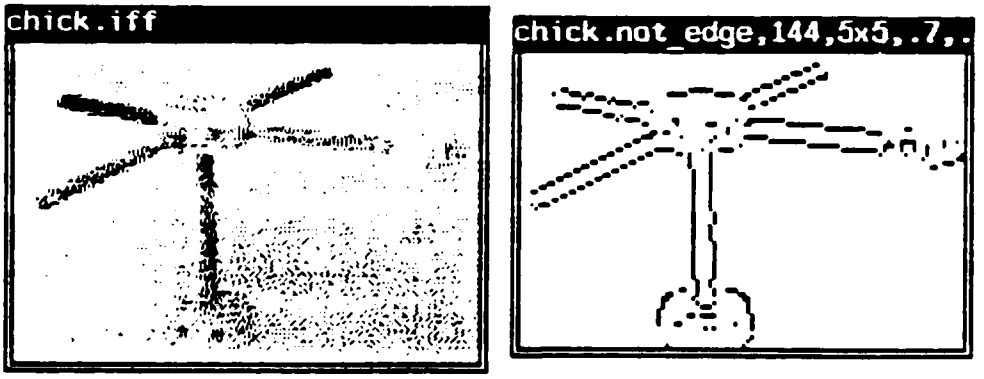
Thus the superiority of the combined operator for false positives dominates the false negative performance and the combined operator minimizes the number of errors in total. These results are evidence that my combination rule is robust.

5.2. Real Images

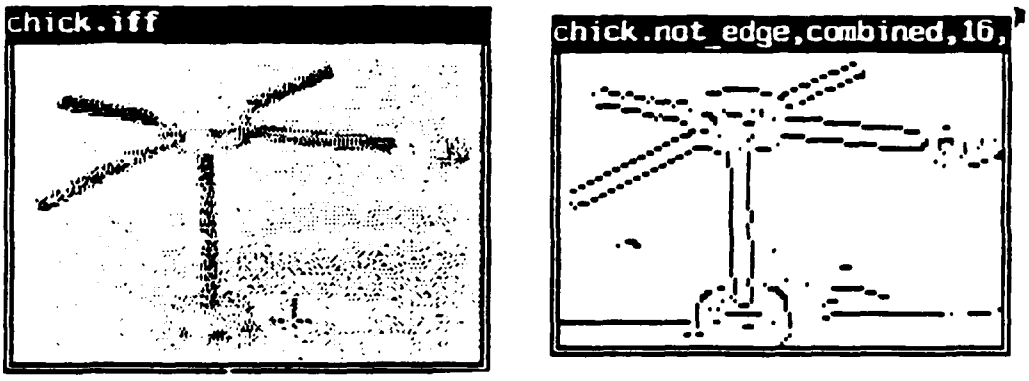
I have also tested these theories using two images taken by cameras. One of these images is a tinker toy image taken in our lab. The other is an aerial image of the vicinity of Lake Ontario. Figure 23 shows the result of the operator tuned to standard deviation 4 noise applied to the tinker toy image and thresholded at 0.5 probability. Figure 24 shows the result of the operator tuned to standard deviation 12 noise applied to the tinker toy image. Figure 25 shows the effect of combining operators tuned to standard deviation 4, 8, 12 and 16 with equal probability.



a: Tinkertoy Image b: Output of $\sigma = 4$ detector
Figure 23: $\sigma = 4$ detector applied to tinkertoy image



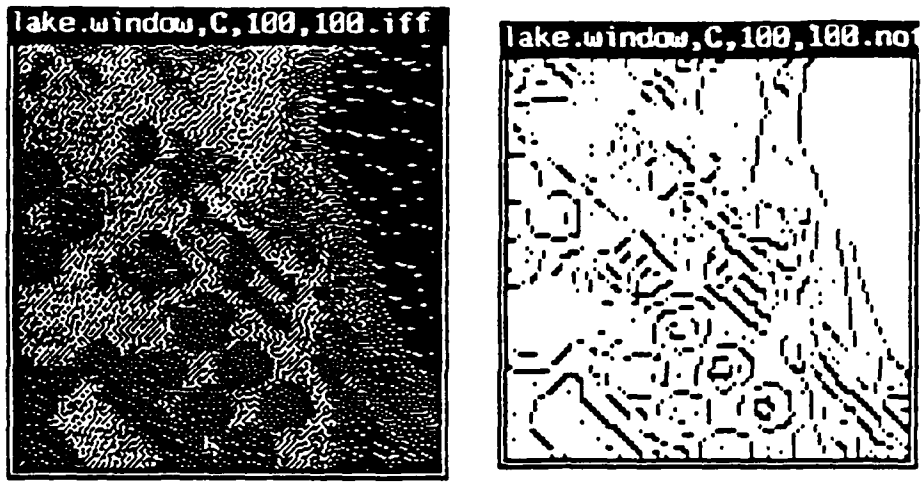
a: Tinkertoy Image b: Output of $\sigma = 12$ detector
Figure 24: $\sigma = 12$ detector applied to tinkertoy image



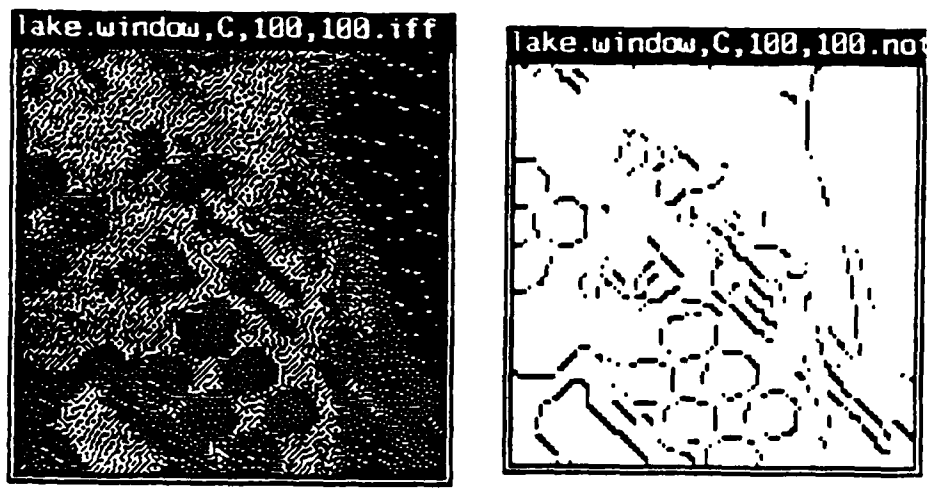
a: Tinkertoy Image b: Output of combined detector
Figure 25: Combined detector applied to tinkertoy image

Here, the result of the combined operator seems to be a cleaned up version of the standard deviation 4 operator. Most of the features that are represented in the output of the combined operator are however real features of the scene. The line running horizontally across the image that the standard deviation 4 operator and the combined operator found is the place where the table meets the curtain behind the tinkertoy. The standard deviation 4 operator was certain of its interpretation and the other operators were uncertain at that point so its interpretation was used by the combination.

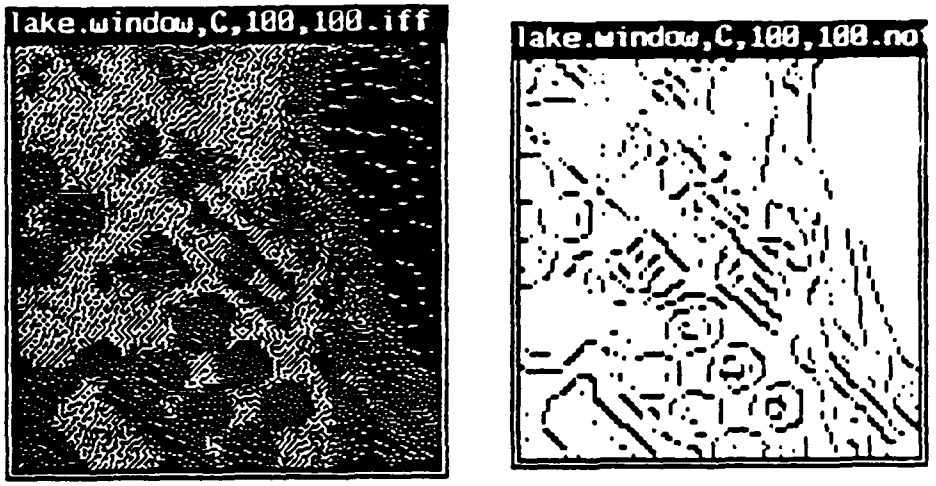
The results from the aerial image are also instructive. Figure 26 shows the result of the operator tuned to standard deviation 4 noise applied to the aerial image and thresholded at 0.5 probability. Figure 27 shows the result of the operator tuned to standard deviation 12 noise applied to the aerial image. Figure 28 shows the effect of combining operators tuned to standard deviation 4, 8, 12 and 16 with equal probability.



a: Aerial Image b: Output of $\sigma = 4$ detector
Figure 26: $\sigma = 4$ detector applied to aerial image



a: Aerial Image b: Output of $\sigma = 12$ detector
Figure 27: $\sigma = 12$ detector applied to aerial image



a: Aerial Image b: Output of combined detector
Figure 28: Combined detector applied to aerial image

The results from the combined operator are again a cleaned up version of the results from the standard deviation 4 operator. I believe this behavior occurs again because the features being found by the standard deviation 4 operator are in the scene. However I do not have the ground truth for the aerial image as I do for the tinkertoy image.

5.3. Future Experiments

Soon, I will apply my evidence combination rules to operators that make different assumptions about the expected image intensity histogram. The operator used so far in my experiments expects a uniform histogram between 0 and 254. Currently, a likelihood generator has been built that assumes a triangular distribution with the probability of an object having intensity less than 128 being one fourth the probability of an object having intensity greater than or equal to 128. It is not clear that the probabilities calculated based on this assumption will be significantly different from those based on the uniform histogram assumption. If there is no difference in the output of two operators the effect of combination is invisible.

Larger operators will soon be available. The likelihoods generated based on these larger operators would be finely tuned. The same evidence combination can be applied to these operators.

Likelihoods are used by Markov random field algorithms to determine posterior probabilities [Marroquin85b] [Chou87]. Likelihoods resulting from my combination rules can be used by Markov random field algorithms.

6. Previous Work

Much of the work on evidence and evidence combination in vision has been on high level vision. An important Bayesian approach (and a motivation for my work) was by Feldman and Yakimovsky [Feldman74]. In this work Feldman and Yakimovsky were studying region merging based on high level constraints. They first tried to find a probability distribution over the labels of a region using characteristics such as mean color or texture. They then tried to improve these distributions using labelings for the neighbors. Then they made merge decisions based on whether

it was sufficiently probable that two adjacent regions were the same.

Work with a similar flavor has been done by Hanson and Riseman. In [Hanson80] Bayesian theories are applied to edge relaxation. This work had serious problems with its models and the fact that the initial probabilities input were edge strengths normalized never to exceed 1. Of course such edge strengths have little relationship to probabilities (a good edge detector tries to be monotonic in its output with probability but that is about as far as it gets). In [Wesley82a] and [Wesley82b] Dempster-Shafer evidence theory is used to model and understand high level problems in vision especially region labeling. In [Wesley82b] there is some informed criticism of Bayesian approaches. In [Reynolds85] They study how one converts low level feature values into input for a Dempster-Shafer evidence system.

In [Levitt85] Tod Levitt takes an approach to managing a hierarchical hypothesis space that is baysian with some ad hoc assumptions. For the problem worked on here the paper would take weighted sums of probabilities. He does not have any way of taking an operators self confidence into account in the evidence combination. Since he was not approaching this problem in his paper I can not fault it in this respect.

There has been much use of likelihoods in recent vision work. In particular work based on Markov random fields [Geman84] [Marroquin85a] [Marroquin85b] use likelihoods. A Markov random field is a prior probability distribution for some feature of an image and the likelihoods are used to compute the marginal posterior probabilities that are used to update the field. Haralick has mentioned that his facet model [Haralick84] [Haralick86b] can be easily used to build edge detectors that return likelihoods [Haralick86a]. I also have built boundary detectors that return likelihoods and the results of using them is documented in [Sher87]. Paul Chou is using the likelihoods I produce with Markov random fields for edge relaxation [Chou87]. He is also studying the use of likelihoods for information fusion. Currently, he is concentrating on information fusion from different sources of information.

7. Conclusion

I have presented a Bayesian technique for information fusion. I show how to fuse information from detectors with different models. I presented results from applying these techniques to artificial and real images.

These techniques take several operators that are tuned to work well when the scene has certain particular properties and get an algorithm that works almost as well as the best of the operators being combined. Since most algorithms available for machine vision are erratic when their assumptions are violated this work can be used to improve the robustness of many algorithms.

References

[Aloimonos85]
J. Aloimonos and P. Chou, Detection of Surface Orientation and Motion from Texture: 1. The Case of Planes, 161, Computer Science Department, University of Rochester, January 1985.

[Ballard82]
D. H. Ballard and C. M. Brown, in *Computer Vision*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1982, 125.

[Chou87] P. Chou, Multi-Modal Segmentation using Markov Random Fields, *Submitted to IJCAI*, January 1987.

[Feldman74]
J. A. Feldman and Y. Yakimovsky, Decision Theory and Artificial Intelligence: I. A Semantics-Based region Analyzer, *Artificial Intelligence 5*(1974), 349-371, North-Holland Publishing Company.

[Frieden85]
B. R. Frieden, Estimating Occurrence Laws with Maximum Probability, and the Transition to Entropic Estimators, in *Maximum-Entropy and Bayesian Methods in Inverse Problems*, C. R. Smith and W. T. G. Jr. (editor), D. Reidel Publishing Company, Lancaster, 1985.

[Geman84]
S. Geman and D. Geman, Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, *PAMI 6*,6 (Novermber 1984), 721-741, IEEE.

[Good50] I. J. Good, *Probability and the Weighing of Evidence*, Hafner Publishing Company., London, New York, 1950

[Good83] I. J. Good, Subjective Probability as the Measure of a Non-measurable Set, in *Good Thinking: The Foundations of Probability and its Applications*, Minneapolis (editor), Univeraity of Minnesota Press, Minneapolis, 1983, 73-82.

[Hanson80]
A. R. Hanson, E. M. Riseman and F. C. Glazer, Edge Relaxation and Boundary Continuity, 80-11, University of Massachusetts at Amherst, Computer and Information Science, May 1980.

[Haralick84]
R. M. Haralick, Digital Step Edges from Zero Crossing of Second Directional Derivatives, *PAMI 6*,1 (January 1984), 58-68, IEEE.

[Haralick86a]
R. Haralick, Personal Communication, June 1986.

[Haralick86b]
R. M. Haralick, The Facet Approach to Gradient Edge Detection, *Tutorial 1 Facet Model Image Processing (CVPR)*, May 1986.

[Horn70] B. K. P. Horn, *Shape from Shading: A Method for FInding the SHape of a Smooth Opaque Object from One View*, Massachusetss Institute of Technology Department of ELectrical Engineering., August 1970

[Ikeuchi80]
K. Ikeuchi, Shape form Regular Patterns (an Example of Constraint Propagation in Vision), 567, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, March 1980.

[Johnson85]
R. W. Johnson and J. E. Shore, Introduction to Minimum-Cross-Entropy Spectral Analysis of Multiple Signals, in *Maximum-Entropy and Bayesian Methods in Inverse Problems*, C. R. Smith and W. T. G. Jr. (editor), D. Reidel Publishing Company, Lancaster, 1985.

[Levitt85] T. S. Levitt, Probabilistic Conflict Resolution in Hierarchical Hypothesis Spaces, *Proceedings: Uncertainty and Probability in Artificial Intelligence*, August 14-16, 1985, 265 272.

[Marroquin85a]
J. Marroquin, S. Mitter and T. Poggio, Probabilistic Solution of Ill-Posed Problems in Computational Vision, *Proceedings: Image Understanding Workshop*, December 1985, 293-309. Sponsored by: Information Processing Techniques Office Defence Advanced Research Projects Agency.

[Marroquin85b]
J. L. Marroquin, Probabilistic Solution of Inverse Problems, Tech. Rep. 860, MIT Artificial Intelligence Laboratory, September 1985.

[Ohlander79]
R. Ohlander, K. Price and D. R. Reddy, Picture Segmentation using a Recursive Region Splitting Method, *CGIP 8*,3 (1979).

[Reynolds85]
G. Reynolds, D. Strahman and N. Lehrer, Converting Feature Values to Evidence, *PROCEEDINGS: IMAGE UNDERSTANDING WORKSHOP*, December 1985, 331-339. Sponsored by: Information Processing Techniques Office, Defence Advanced Research Projects Agency.

[Sher86] D. Sher, Optimal Likelihood Detectors for Boundary Detection Under Gaussian Additive Noise, *IEEE Conference on Computer Vision and Pattern Recognition*, Miami, Florida, June 1986.

[Sher87] D. B. Sher, Advanced Likelihood Generators for Boundary Detection, TR197, University of Rochester Computer Science Department, London, England, January 1987. Submitted in shorter form to International Conference on Computer Vision.

[Wesley82a]

L. P. Wesley and A. R. Hanson, The Use of an Evidential-Based Model for Representing Knowledge and Reasoning about Images in the Visions System, *PAMI* 4,5 (Sept 1982), 14-25, IEEE.

[Wesley82b]

L. P. Wesley and A. R. Hanson, The use of an Evidential-Based Model for Representing Knowledge and Reasoning about Images in the VISIONS System, *Proceedings of the Worksshop on Computer VIsion: Representation and Control*, August 1982, 14-25.

Advanced Likelihood Generators for Boundary Detection

David Sher
Computer Science Department
The University of Rochester
Rochester, New York 14627

January 1987
TR 197

In [Sher85] I discussed the advantages of feature detectors that return likelihoods. In [Sher86] I demonstrated some preliminary results with a boundary point detector that returns likelihoods. Now I have developed boundary point detectors that have these properties:

- Return probabilities
- Can be combined robustly
- Potential sub-pixel precision
- Work with correlated noise.
- Can handle multiple gray-levels (tested with 255)

These detectors were applied both to aerial images and to test images whose boundaries are known. They are compared to established edge detectors.

This work would have been difficult if not impossible without the active support and encouragement of Chris Brown, my advisor, and Jerry Feldman. This work was supported by the Defense Advanced Research Projects Agency U. S. Army Engineering Topographic Labs under grant number DACA76-85-C-0001. Also this work was supported by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, and the Air Force Office of Scientific Research, Bolling AFB, DC 20332, under Contract No. F30602-85-C-0008. This contract supports the Northeast Artificial Intelligence Consortium (NAIC).

1. Introduction

Currently a great variety of tools are available for low-level vision tasks such as image reconstruction and edge detection. It is time to devote attention to managing tools rather than creating new ones.

Most of the tools for low level vision are algorithms for intermediate steps towards achieving a goal. Here, we consider boundary point detection algorithms in these terms. These are algorithms that try to determine if a boundary passes through a pixel (usually given a window on the image). This task is similar to edge detection. Boundary point detection algorithms do not exist to display outlines pleasing to the human eyes. Their output is meant to be input to a higher level routine such as a shape recognition program or a surface reconstruction program.

Some desiderata for boundary point detection tools are:

(1) The output of a boundary point detector should be useful to as many higher level routines as possible. If every higher level routine required a different boundary point detector then our technology has not lived up to this desiderata.

(2) Boundary point detectors should accept as input the full range of data available. If a boundary point detector only worked for binary data when gray scale data was available, it has not lived up to this desiderata.

(3) Boundary point detectors should do work proportional to the size of the image.

(4) Boundary point detectors should do work proportional to the precision of the output. Thus if subpixel precision is required, it should be made available with work proportional to the required accuracy of boundaries reports.

(5) Boundary point detectors should be parameterized to features of the data. For example, if the distribution of reflectances in the scene is known (the expected image histogram) then a detector should be constructed that uses this information. Another example is if structure in the noise is known (such as correlation) we should be able to take this structure into account.

In this paper I describe an algorithm that fits these desiderata. It is a more advanced version of the algorithm described in [Sher86]. Also the results of tests run on this algorithm are reported here and comparisons with established algorithms such as the Sobel, Kirsch and variants thereupon. Tests will be done soon on more sophisticated operators.

2. Definition of Boundary

Before talking about boundary point detector it is a good idea to define exactly what a boundary is. Vision problems involve imaging a scene. This scene could be an aerial view or a picture of machine parts or an outdoors scene. This scene is filled with *objects*. In an aerial photograph some of these objects are buildings, trees, roads and cars. Each of these objects is projected into the observed image. Thus each object has an *image* that is a subset of the observed image. Where the observed image of one object meets an observed image of another object there is a *boundary*. such boundaries are sometimes referred to as occlusion boundaries.

3. Returning Probabilities

The algorithm I describe fulfills the first desideratum by returning probabilities that a boundary is near a point. Here I justify returning probabilities and show how I can fulfill the first

desideratum using them.

No tool for boundary point detection or any other low-level vision task ever does its task without a significant probability of being wrong. Thus the error characteristics of a low-level vision algorithm need to be considered. Intermediate-level vision algorithms differ in sensitivities to different kinds of errors.

Consider boundary point detection. Regularization algorithms (interpolation algorithms) [Boult6?] suffer more from missing a boundary point than from having extra ones. When a boundary point is missed regularization algorithms try to smooth over the boundary with disastrous results. Hough transform techniques often work effectively when the set of boundary points detected is sparse because the work a hough transform technique does is proportional to the size of that set.

Another reason that Hough transform techniques do well with sparse data is that they are mode based. Thus Hough transform techniques are robust when feature points are left out and when there are outlying data points. The robustness of the Hough transform is described in detail in [Brown82].

It is good software management to use the same boundary point detector to generate input for all high level vision tasks that require boundary point detection rather than building a special detector for each high level routine. If a boundary point detector returns a true/false decision for each point then its output does not suit both regularization techniques and hough transform techniques. Take for example the one dimensional intensity slice shown in Figure 1.

Figure 1: Slice through an image with an ambiguous boundary

For use as a first stage before regularization it is preferable that such ambiguous slices be considered boundaries because the cost of missing a boundary is high (compared to the cost of missing an edge). For use with hough transform line detection figure 1 is not a good boundary because the cost of an extra edge is high.

The traditional solution to the dilemma of satisfying differing requirements among intermediate level routines has been to supply numbers such as *edge strengths* rather than true/false decisions. These strengths describe how likely the low-level vision algorithm considers the event such as the existence of a boundary. The example in figure 1 has the detector return a low edge strength. Figure 2 shows a 0 edge strength.

Figure 2: Slice through an image with an edge strength of 0

Figure 3 shows a high edge strength.

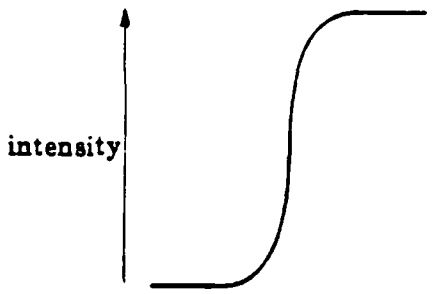


Figure 3: Slice through an image with a high edge strength

If an edge detector that returns strengths (acting as a boundary point detector) is used as input for various intermediate level applications, then each application usually uses a threshold to determine what is and isn't a boundary. If an edge strength is higher than the threshold a boundary is reported. The threshold is determined by running the boundary point detector and the intermediate level algorithm with different thresholds and finding the threshold that results in the best results according to some error norm or human observation[1]. If the boundary point detector is changed new thresholds need to be found.

If boundary point detectors were standardized so that the correspondence between strength and the probability of the boundary were consistent between all boundary point detectors then the threshold need only be calculated once for each intermediate level application. Then the thresholds for intermediate level applications could be calculated from theoretical principles. When the strengths have clear semantics the entire process of threshold determination would be fully understood.

A consistent and well defined output for boundary point detectors is the probability of a boundary. If all boundary point detectors output the probability of a boundary then the boundary point detector could be improved without changing the rest of the system. If the error sensitivities of the intermediate level routines are known the thresholds can be determined by a simple application of decision theory.

4. Likelihoods

Most models for boundary point detection describe how configurations of objects in the real world generate observed data. Such models do not explicitly state how to derive the configuration of the real world from the sensor data. This behavior of models results in graphics problems being considerably easier than vision problems. Thus we have programs that can generate realistic images that no program can analyze.

Given the assumption "There is a boundary between pixels p_1 and p_2" we can determine a probability distribution over possible observed images. Let $b(1,2)$ be the statement that there is a boundary between pixels p_1 and p_2. Let $S(b(1,2))$ be the set of region maps such that $b(1,2)$ is true (and generally I use the notation that $S(statement)$ is the set of region maps where *statement* is true). Let M be the model for the boundary detection task. Then the probability of O (the observed image) given $b(1,2)$ under M is calculated by equation 1.

[1]Edge relaxation algorithms often adjust the edge strengths. For the purposes of this paper consider such an algorithm as a part of the detector, the output of this detector is the strengths output by the relaxation algorithm.

$$P(O|b(1,2)\&M) = \frac{\sum_{l \in S(b(1,2))} P(O|l\&M)P(l|M)}{P(b(1,2)|M)} \quad (1)$$

If O is the observed data then the probability calculated in equation 1, $P(O|b(1,2)\&M)$, is called here (inspired by the statistical literature) the *likelihood* of $b(1,2)$ given observed data O under M. Generally the models we use in computer vision make the calculation of likelihoods simple for the features we want to extract. However the desired output of a feature detector is the conditional probability of the feature. Thus in boundary point detection I can derive $P(O|b(1,2)\&M)$ and I can derive $P(O|\sim b(1,2)\&M)$ ($\sim x$ means "not x" in this paper) but I want to derive $P(b(1,2)|O\&M)$.

A theorem of probability theory, *Bayes' law*, shows how to derive conditional probabilities for features from likelihoods and prior probabilities. Bayes' law is shown in equation 2.

$$P(f|O\&M) = \frac{P(O|f\&M)P(f|M)}{P(O|f\&M)P(f|M)+P(O|\sim f\&M)P(\sim f|M)} \quad (2)$$

In equation 2 f is the feature for which we have likelihoods. M is the model we are using. $P(O|f\&M)$ is the likelihood of f under M and $P(f|M)$ is the probability under M of f (the prior probability).

For features that can take on several mutually exclusive labels such as surface orientation a more complex form of Bayes' law shown in equation 3 yields conditional probabilities from likelihoods and priors.

$$P(l|O\&M) = \frac{P(O|l\&M)P(l|M)}{\sum_{l' \in L(f)} P(O|l'\&M)P(l'|M)} \quad (3)$$

l is a label for feature f and $L(f)$ is the set of all possible labels for feature f.

Likelihoods are important because they are a useful intermediate term on the way to deriving a posterior probability. An upcoming technical report describes formulas for evidence combination based on likelihoods [Sher87]. To apply that theory of evidence combination one needs to compute the likelihoods explicitly. Thus in the succeeding sections I calculate the likelihoods even up to factors that are equal in all the likelihoods (hence are divided out by equation 3).

Another important use for explicit likelihoods is for use in Markov random fields. Markov random fields describe complex priors that can capture important information. Markov random fields were applied to vision problems in [Geman84]. Likelihoods can be used with a Markov random field algorithm to derive estimates of boundary positions [Marroquin85] [Chou87].

5. Simplifications

To make the problem of computing the likelihoods for events computationally tractable, I simplify my problem in certain ways.

The first simplification is calculating the probability of a boundary at a point rather than trying to compute a probability distribution over boundary maps for the entire scene. Even though a distribution over complete maps would be more general there are too many possible boundary maps to manage realistically. Under certain circumstances all that is needed is to compute the probability of a boundary near each pixel [Marroquin85].

5.1. Window Based Boundary Detectors

Equation 1 implies an algorithm for determining likelihoods of boundaries. It shows that iterating through all the configurations that cause a boundary at a point is a way to find the likelihood of a boundary. A *region map* is a description of where the images of the objects are. Each configuration of objects in a scene implies a region map. However, the set of region maps that contain a boundary between two pixels is too large to iterate through (for a 512 by 512 observed image it is $>>10^{25,000}$). The cardinality of the set of region maps is in general exponential in the size of the image.

An obvious solution to the problem is to reduce the number of pixels. Generally, the further one gets from a boundary the less relevant the data in the image is to that boundary. Thus it is common to use a relatively small window about the proposed boundary for finding the boundary (see figure 4). There are many fewer region maps over a 4 by 4 window than over a 512 by 512 image.

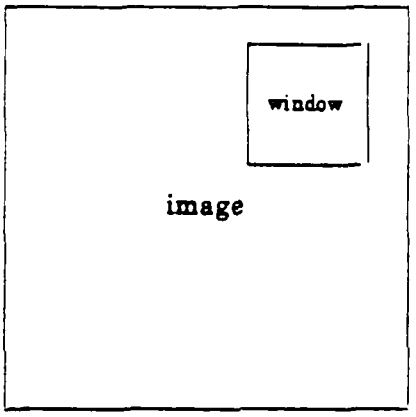


Figure 4: Small Window on Large Image

Thus by looking only at a window I simplify the problem of boundary point detection considerably. Inevitably, I lose some accuracy in the computation of probabilities from limiting the data to a window. However the simplification of the algorithm compensates for this loss. Every attempt at edge detection has used this principle [Hueckel71] [Canny83] with possibly a further stage of linking or relaxation.

5.2. Constraining Region Maps

Applying a boundary point detector to an H (height) by D (depth) observed image involves computing the probability of HD boundaries. If $H=D=512$, then $HD=262144$. A boundary point detection algorithm computes probabilities for all these potential boundaries. The cost of using algorithm that computes the probability of a boundary at a point is multiplied by HD. The cost may be reduced by sharing some of the work between iterations. Still much of the work can not be shared. Saving time by sharing work between iterations is discussed in later sections.

Ignoring saving by sharing between iterations, any saving in the algorithm that computes the probability of a boundary at one point is multiplied manyfold (for $H=D=512$ 262144fold). Algorithms for computing the probability of a boundary at a point require work proportional to the number of region maps. Reducing the number of region maps that need to be considered proportionately reduces the work required by the algorithms for boundary point detection.

The maximal amount a region map can change the likelihood of a feature label is the probability of that region map divided by the prior probability of a feature label corresponding to that region map (shown by a cursory examination of equation 1, repeated below, with i being a region map).

$$P(O|b(1,2)\&M) = \frac{\sum_{i \in S(b(1,2))} P(O|i\&M)P(i|M)}{P(b(1,2)|M)} \quad (1)$$

Bayes' law (equation 3) can be broken into two steps. First the likelihoods are multiplied by the priors. Consider the likelihood of a feature label (say *boundary*) multiplied by the prior probability. Call such a term the *conjunctive probability* of l since it is $P(O\&f=l\&M)$. The conjunctive probability of l can be changed by deleting a region map with $f=l$ at most the probability of that region map. The probability $P(f=l|O\&M)$ is derived from the conjunctive probabilities by dividing $P(f=l\&O\&M)$ by the sum of the conjunctive probabilities $\sum_r P(f=r\&O\&M)$. Thus if the probability of a region map is small compared to $P(O\&M)=\sum_r P(f=r\&O\&M)$ deleting the likelihood corresponding to it has a small effect on the resulting distribution. Thus one can with some safety ignore region maps whose prior probability is small enough.

For standard edge detection algorithms a common restriction on region maps is to assume that there are at most two object images participating in the window thus there can only be two regions [Hueckel71]. Step edge based models implicitly make this assumption [Canny83] [Nalwa84]. Windows with more than two object images in them are assumed to occur infrequently I call the assumption that there are at most two object images participating in a window the *two object assumption*.

Another simplification places a limitation on the curvature of the boundaries observed in the image. Limiting this parameter limits the set of region maps in a mathematically convenient way. If the curvature is limited enough the boundaries can be considered to be straight lines within windows. Thus the windows on the observed image can be modeled assuming there is no boundary or a single linear boundary across them. A similar model (it allowed two linear boundaries in a window) was used in [Hueckel71]. I call the assumption that there are no high curvature boundaries the *low curvature assumption*.

5.3. Numerical Approximations

Another effect that makes the probabilities calculated by my algorithms inaccurate is that a real number can only be specified to a limited accuracy on the computer. Thus there is a limit to the accuracy that calculations can be performed to in the computer. In section 6 I ignore the error introduced from inexact floating point computations. In my implementation (section 8) I used double precision arithmetic throughout in an attempt to reduce this error.

Another source of errors is my simplifying the mathematics to make the algorithm simpler. One such approximation I make is to use the density of a normal distribution as a probability. In the equations derived in section 6 I use this approximation in the probability derived from a multinormal Gaussian. This approximation simplified the mathematics for deriving the detector. Such an approximation is standard

6. Building a Boundary Detector

In section 4 I discuss why determining likelihoods is a useful first step in feature detection. In section 5 I describe the approximations and simplifications necessary to make the problem of boundary point detection computationally tractable. Now all that is left is to develop the algorithm. The first step in deriving a boundary point detection algorithm is to derive the likelihood that a window is filled with a single object given the observed data (section 6.1). Then likelihoods for windows with multiple objects are derived (section 6.3). In this section I assume that the model has Gaussian mean 0 noise with a known standard deviation added to an ideal image.

6.1. Likelihoods for a Single Object

The problem is to find the likelihood of a single object filling a window in the image. The expected intensities of the pixels in the window are proportional to the reflectance of O. Let T_0 be the expected value of the pixels in the window when O has reflectance 1, $r(O)=1$. T_0 is often referred to as a *template* in the image understanding literature.

Template matching can be best shown on a two dimensional medium (with weak graphical capacities) when the window is 1 dimensional. Thus I use a 1 by 8 window for examples. A typical template for a single object in a 1 by 8 window is shown in figure 5.

100	100	100	100	100	100	100	100

Figure 5: Template for a single object

This template is boring because I assume that there is little variance in intensity in the image of the interior of an object. The observed image of the object has a normal iid added to each pixel. Thus the observed image (when the standard deviation is 8) can look like figure 6.

94	104	100	94	92	105	101	103

Figure 6: Noised Template for a single object

The probability that the noised window results from the template is a function of the vector difference between the window and the template. For the example in figure 6 given the template for a single object in figure 5, the probability is calculated by summing the squared differences between the template and the observed data (187) and then applying the normal distribution function (for 8 independent normally distributed samples mean 0 standard deviation 8 rounded to the nearest integer) to get 8.873e-12.

In later sections when I use windows or templates in equations the windows or templates are flattened into vectors. Thus a two dimensional window is transformed into a vector (figure 7).

1	2	3
4	5	6
7	8	9

Window 1

1	2	3	4	5	6	7	8	9

The Vector from Window 1.

Figure 7: Flattening a window into a vector

All the windows and templates are assumed to be flattened thus. When a template is subtracted

from a window, vector subtraction is happening. When a template or window is being multiplied by a matrix, vector matrix multiplication is happening. In particular, WW^T is the sum of the squares of the elements of W while WT^T is the sum of the products of the corresponding elements of W and T.

Let aT_0 be a times the intensities in the template T_0. The template for an object with reflectance $r(O)$ is $r(O)T_0$. The probability of observing a window W when the scene is of O is determined by the Gaussian additive factor. Equation 4 is the formula for the probability. (Let V be the variance of the noise, σ^2 in the rest of this paper) Let n be the size of the window and κ be the constant $1/(2\pi V)^{n/2}$.

$$P(W|r(O)=r\&M)=\kappa\exp(-(W-rT_0)(W-rT_0)^T/2V) \tag{4}$$

Since the reflectance of the object in the scene is not known but the distribution of reflectances is known one must integrate this formula over possible reflectances. Thus the probability of a window given it is of a single object of known position and shape is in equation 5.

$$P(W|O\&M) = \int\kappa\exp(-(W-rT_0)(W-rT_0)^T/2V)dD_r(r) \tag{5}$$

The term $(W-rT_0)(W-rT_0)^T$ can be rearranged to $WW^T-2rT_0W^T+r^2T_0T_0^T$. WW^T is the sum of the squares of the pixels in the window. Let us refer to it as W^2 for shorthand. T_0W^T is the correlation between T_0 and W. Let us refer to it as C for shorthand. $T_0T_0^T$ is the sum of the squares of the elements of the template. Let us refer to it as T^2. Using a rearrangement and these shorthands equation 5 can be restated as equation 6.

$$P(W|O\&M) = \exp(-W^2/2V)\kappa\int\exp((2rC-r^2T^2)/2V)dD_r(r) \tag{6}$$

Equation 6 is the product of two parts, equation 7 :

$$F_1(W^2) = \exp(-W^2/2V)\kappa \tag{7}$$

and equation 8 :

$$F_2(C) = \int\exp((2rC-r^2T^2)/2V)dD_r(r) \tag{8}$$

F_1 and F_2 have one parameter that depends on the window (T^2 is unchanged over all windows). They can be implemented by table lookup without excessive use of memory or much computation.

Hence the algorithm to calculate the likelihood of a window given it is of a single object of known position and shape (but unknown reflectance) is described by figure 8 below.

Figure 8: Algorithm to Calculate Likelihood of Known Object for a Single Window

			Multiplies	Adds
W^2	:=	WW^T	w	$w-1$
C	:=	WT_0^T	w	$w-1$
Output		$F_1(W^2)*F_2(C)$	1	0

w is the number of pixels in W and also the number of pixels T_0. Figure 8 is a $4w-1$ operations algorithm.

6.2. Reducing the Configurations Considered

In section 4 I describe how to derive the likelihood of a boundary at a point from the likelihoods of configurations of a the scene given a boundary at a point (equation 1). Here I justify

using a small number of configurations of the scene. Having reduced the set of configurations I can derive an efficient algorithm for generating likelihoods given multiple objects. In section 5.1 I made the assumption that only a small window on the image need be looked at. In section 5.2 I justified ignoring region maps with low probability.

There is some number of objects, N_O, such that the probability N_O or more objects having an image in a single window has a probability much smaller than the probability of all but a small probability subset of the region maps. Hence I need only consider configurations of $N_O - 1$ or less objects in a window. The two object assumption of section 5.2 is equivalent to saying $N_O = 3$.

There are still a large set of configurations of less than N_O objects. Consider the ideal image given such a configuration and some coloring of objects. Consider a window on this ideal image, W_I. There is a set of configurations with the same coloring such that the resulting window on the ideal window from such a configuration W_J such that:

$$(W_I - W_J)(W_I - W_J)^T < \epsilon \tag{9}$$

The likelihood that the observed image results from any of the configurations that fit the criterion of equation 9 and coloring is close to the likelihood computed from W_I because if W is the observed window the likelihood is a function of the norm of $W - W_J$. Thus for efficiency sake we can consider the likelihood of each configuration and coloring that fit equation 9 the same as that of the configuration and coloring of the template that generates W_I. Hence we can only consider a small set of configurations of objects. With a similar argument one can prove that only a subset of the possible colorings need be considered. Hence I can justify using an argument of this sort using a small set of templates and objects. How much inaccuracy results from these simplifications can be analyzed mathematically.

A step edge model can be derived by assuming that objects' images have a uniform intensity and the two object assumption. Thus such simplifications underlies work like that of [Hueckel73] [Canny83]. More sophisticated assumptions about the intensities of objects images result in more complex models [Binford81] that still can be reduced to a reasonable number of configurations reflectances with a corresponding loss of accuracy in the resulting probabilities.

6.3. Algorithm for Likelihoods of Multiple Objects

Here I derive an algorithm for finding the likelihood that a scene that contains several objects given a window.

The statement that the configuration of objects is near the specified configuration is called C. C has N_O objects O_n with reflectances r_n. Associated with C are N_O templates T_n. Each T_n is the light that hits the image given that O_n has reflectance 1 and unit lighting.

The template that represents the expected window when the O_n have reflectances r_n is $\sum_n r_n T_n$. Hence the likelihood of the window when the objects have reflectances r_n is shown in equation 10.

$$P(W|\mathbf{C}\&r(O_1)=r_1 \cdots r(O_n)=r_n\&M) = \kappa \exp(-(W - \sum_n r_n T_n)(W - \sum_n r_n T_n)^T/2V) \tag{10}$$

To derive the likelihood of the window when the objects are of unknown reflectance it is necessary to integrate over all possible sets of r_n. Equation 11 shows the integrated equation.

$$P(W|C\&M)$$

$$= \tag{11}$$

$$\kappa \int \exp(-(W - \sum_n r_n T_n)(W - \sum_n r_n T_n)^T/2V) dD_e r_n$$

I use the simplifying notation from section 6.1. Also let C_n be WT_n^T. Then equation 11 can be transformed into equation 12.

$$P(W|C\&M)$$

$$= \tag{12}$$

$$\kappa \exp(-W^2 + w\alpha^2/2V) \int \exp(\sum_n r_n C_n/V) \exp(-(\sum_n r_n T_n)(\sum_n r_n T_n)^T/2V) dD_e r_n$$

As in section 6.1 I can break up equation 12 into a pair of functions F_1 and F_4. F_1 is as before F_4 is described by equation 13.

$$F_4(C_1, C_2, \cdots C_{N_O}) = \int \exp(\sum_n r_n C_n/V) \exp(-(\sum_n r_n T_n)(\sum_n r_n T_n)^T/2V) dD_e r_n \tag{13}$$

Unlike F_2, F_4 is a function of several variables. Thus using a table for table lookup on F_4 takes a large amount of memory since an entry must be made available for each possible value of the elements. Experiments with constructing F_4 tables for 2 object configurations show that F_4 is smooth and near quadratic. As an example F_4 for standard deviation 12 noise and 5 by 5 templates is plotted in figure 9.

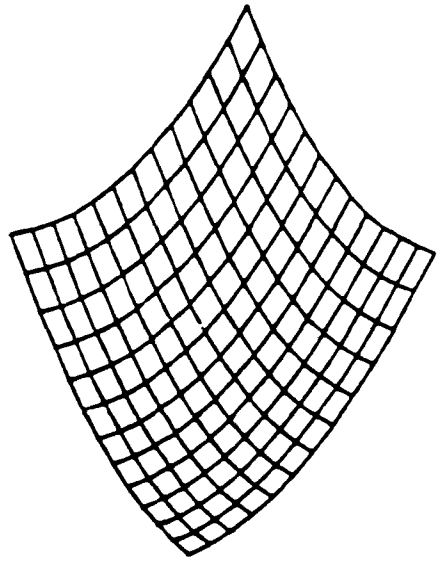


Figure 9: $\log(F_4)$ with stdev 12 noise and 5x5 template

Hence the tables can be stored sparsely and splines or even linear interpolation used to get at values that were not given without introducing serious inaccuracy. If the table is close enough to a quadratic function then perhaps only a polynomial need be stored.

If all the T_n are orthogonal (their pairwise vector products are 0) and D_e is a probability distribution where colors of objects are uncorrelated then F_4 can be partitioned into a product of N_O functions. Each of these functions computes the probability that the relevant part of the window observed data is the image of O_n for some n. Here N_O large precise tables can be stored to compute each of the functions and their product used for F_4.

Since F_4 can be calculated there is a feasible algorithm for determining the likelihood of an observed image given a particular pair of objects. A description of that algorithm and the times spent in each step is shown in figure 10.

Figure 10: Algorithm to Calculate Likelihood of Multiple Objects for a Single Window

			Multiplies	Adds
W^2	:=	WW^T	w	$w-1$
N_O times C_i	:=	WT_i^T	$N_O w$	$N_O w - N_O$
Output		$F_1(W^2)*F_4(C_1,C_2)$	1	0

I did not consider the cost of the interpolation or splining for F_4 in the operation counts in figure 10. The cost of this algorithm is $(N_O+1)w+1$ multiplies and $(N_O+1)(w-1)$ adds plus the cost incurred by the interpolation. This cost occurs for each window.

6.4. Blurred Images

The previous sections assume that the only degradation of the image data is a result of adding a normal random variable to each element of the image (noise axiom). However lenses and many other sensors degrade the image through blur. Motion also causes blur on film and other sensors. Blur is often modeled as a linear transformation of the image data that is applied before the normal variable is added to the pixels [Andrews77]. When blur is shift independent (same blur everywhere in the image) then the linear operator must be a convolution operator.

The algorithms specified in sections 6.1 and 6.3 require changes to work under this new assumption. If the blur is linear the change required to these algorithms is to use blurred templates. The likelihood of the observed data given a template is a function of the vector difference between the expected observation without the normal additive iid and the observed data. If it is expected that a blurring has happened then one need to use a blurred template instead of the unblurred template used in the previous algorithms.

The algorithm of section 6.3 uses two templates (one for each object). If the blur is linear then the linear function aT_1+bT_2 and the blur function commute. Thus the two templates can be blurred and the result of correlating the window with the two blurred templates can be used with this algorithm.

The problem I address in the rest of this section is how to compute a blurred template from an unblurred template. A shift independent blurring operation is applying a convolution operator to the image. A convolution operator has limited extent if the illuminance of at a pixel in the blurred image depends on a window in the unblurred image. Such a convolution operator can be described by a matrix the size of the window that describes the effect each point in the window has on that point of the blurred image. So a blurring function that causes each point of the blurred image to be the result of .5 from the corresponding point of the unblurred image and .25 from the points immediately to the right and the left has a matrix described by figure 11.

.25	.5	.25

Figure 11: Simple Blurring Function Matrix

Given a blurring function matrix of size (M_w, M_l) and an unblurred template of size (T_w, T_l), a blurred template of size $(T_w - M_w + 1, T_l - M_l + 1)$ can be calculated (figure 12) ($\otimes$ means convolution).

T:

100	100	100	200	200	200

$\otimes$

M:

.25	.5	.25

$=$

$T \otimes M$:

100	125	175	200

Figure 12: Effect of Blur Matrix M on Template T

To develop a larger blurred template than $(T_w - M_w + 1, T_l - M_l + 1)$ requires that the blur function be applied to points outside the unblurred template. If the expected values for such points are derived then a larger unblurred template has been constructed. Hence the derivation of a smaller blurred template from an unblurred template suffices for the construction of blurred templates.

6.5. Correlated Noise

The previous sections assume that an uncorrelated normal variable called noise that is added into the illuminance of each pixel in the ideal image to get the observed image. It is possible to relax the assumption that the noise added to each pixel is uncorrelated with the noise added to the other pixels. Instead a matrix C can be supplied that describes the correlation between the noise variables of the window.

One problem is how to handle correlations between points in the window and points outside the window. Since one can only correlate with expected values of points outside the window (since we chose to ignore the data from such points in our calculations) the effect of such points can only introduce a constant factor into the likelihood calculations. When the likelihoods are converted into probabilities this constant factor is divided out. Hence I can safely ignore such a constant factor. For the purposes of evidence theory I may need to derive the constant factor but it need only be derived once and then all the likelihoods be only multiplied by it.

The algorithm in section 6.3 has the algorithm in sections 6.1 as a special case. Thus if I derive the algorithm corresponding to the one in section 6.3 I can derive the other algorithm. If I have window W and I expect (possibly blurred) templates T_n with unknown reflectances then the equation that describes the likelihood of W is equation 14.

$$P(W|O_1 \cdots O_n \& M) = \kappa \int \exp(-(W - \sum_n r_n T_n) C (W - \sum_n r_n T_n)^T / 2) dD_e r_n \quad (14)$$

I introduce notation to simplify equation 14 to the point where an algorithm naturally derives from it. Let W_C^2 be WCW^T. C is symmetric so let $c_n = WCT_n^T = T_1 CW^T$. Then equation 14 can be rearranged and simplified to equation 15.

$$P(W|O_1 \cdots O_n \& M) = \kappa \exp(-W_C^2/2) \int \exp(\sum_n r_n c_n) \exp(-(\sum_n r_n T_n) C (\sum_n r_n T_n)^T / 2) dD_e r_n \quad (15)$$

I can then describe equation 15 as the product of two functions. F_5 that takes W_C^2 as an argument and F_6 that takes the set of c_n as arguments. Equations 16 describe F_5 and F_6.

$$\begin{aligned} F_5(X) &= \kappa \exp(-X/2) \\ F_6(X_1, \cdots X_{N_O}) &= \int \exp(\sum_n r_n X_n) \exp(-(\sum_n r_n T_n) C (\sum_n r_n T_n)^T / 2) dD_e r_n \\ P(W|O_1, \cdots O_{N_O} \& M) &= F_5(W_C^2) F_6(c_1, \cdots c_{N_O}) \end{aligned} \quad (16)$$

Equation 16 is simple enough to derive an algorithm that calculates the likelihood of a window given a template and correlated noise with standard deviation σ and correlation matrix C. Figure 13 shows this algorithm.

Figure 13: Algorithm to Calculate Likelihood of Multiple Objects with Correlated Noise

			Multiplies	Adds
W_C^2	:=	WCW^T	$w(w+1)$	$(w+1)(w-1)$
N_O times c_2	:=	WCT_n^T	$N_O w$	$N_O(w-1)$
Output		$F_5(W_C^2)*F_6(c_1,c_2)$	1	0

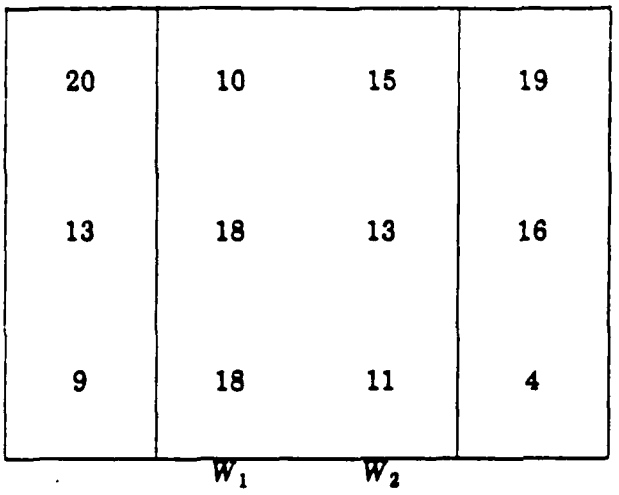
Like F_4, F_6 may require interpolation. The cost of the potential interpolation was not figured into these calculations. The algorithm with correlation between the noise variables requires $w(w+1)+N_O w+1$ multiplies and $(w+1)(w-1)+N_O(w-1)$ adds. Substantial savings may be found when C is sparse. Correlation matrices are typically band matrices. If there are b bands in C then the number of multiplies is less than $(b+1)w+N_O w+1$ and the number of adds is less than $(b+1)(w-1)+N_O(w-1)$ adds.

6.6. Sharing the Work

The algorithms in figures 8, 10 and 13 are algorithms for finding the likelihood of a particular template for a single window. Many of an observed image's windows overlap. If likelihoods are being computed for two overlapping windows much of the work in computing the likelihoods can be shared between the computations on the two windows. If the likelihoods are being computed for every window on the image such savings can be substantial.

When taking the sum of the elements of two overlapping windows, as is one in the algorithm of figures 8 it is necessary to only sum the overlap once. Figure 14 gives an example of this

savings.



$$\Sigma(W_1 \cap W_2) = 10+15+18+13+18+11 = 85$$

$$\Sigma(W_1) = 20+13+9+\Sigma(W_1 \cap W_2) = 127$$

$$\Sigma(W_2) = \Sigma(W_1 \cap W_2)+19+16+4 = 124$$

Figure 14: Summing the elements of two overlapping windows

The work in summing the squares of the elements in two windows can be shared this way too. If the likelihood generator is being used on every window on an image then the work needed to calculate sums and sum of squares is a fraction of that needed to calculate the same statistics for the same number of non-overlapping windows.

If every window (or a substantial fraction thereof) of an image has the algorithms in figure 8 or 10 run on them then the work involved in convolving the image with templates can be saved too (at least when the templates grow large). Convolution can be performed with the fast Fourier transform at substantial saving in operations for large templates. For algorithm 13 when the correlation matrix has structure (such as being a band matrix) then the fast Fourier transform can be used with substantial savings too.

Thus much of the work can be shared when likelihoods are being determined for every window of an image. Hence the likelihood generators described in figures 8, 10, and 13 are competitive in speed with most standard edge detections schemes.

Another way the work can be shared is that some of the templates used that describe object configurations in a window is by describing the configuration in another template shifted 1 pixel over (see figure 15).

Figure 15: T_1 is T_2 shifted 1 pixel to the right

If every window in the image is being processed then the likelihoods corresponding to the template T_2 are approximated by the likelihoods calculated by the template T_1 for the window 1 pixel to the right. To realize why such an approximation is good, consider that using a window is itself an

approximation. The likelihood of the configuration of objects described by T_1 is approximated by running the algorithm over a window. The likelihood of the configuration described by T_2 can be approximated by running the same algorithm over a window shifted 1 to the right.

Thus the likelihoods for the template corresponding to T_2 need only be calculated for the windows on the far right hand side of the image (the other windows have the T_1 template run on them already). Thus instead of having to take into account templates corresponding to the same configuration of objects shifted several pixels in some direction one need only use a single template and use the output from this template on windows shifted in that direction.

6.7. Getting Probabilities from Likelihoods

Given likelihood generators the remaining task is to calculate probabilities from these likelihoods (using priors). The first task that needs to be done is to group the likelihoods generated into sets that support different labelings for the features. Thus if the configurations C_1, C_2, C_3 correspond to the existence of a boundary at a point and C_4, C_5 and C_6 represent situations that aren't boundaries at that point then I must collect the likelihoods based on C_1, C_2 and C_3 into a single likelihood and similar with the likelihoods collected from C_4, C_5 and C_6. Then I would have a likelihood corresponding to each possible labelings of my feature (for boundary point detection I need to determine the likelihoods corresponding to the existence of a boundary and those that correspond to the nonexistence). Given these likelihoods I can use Bayes' rule to derive probabilities (see equation 3).

The likelihood of a boundary is the probability of the observed scene being generated when an object configuration corresponding to the existence of a boundary exists. I can derive an equation for calculating the probability of a boundary from the outputs of my likelihood generators if I have the prior probability that the configuration is the position of the objects in the scene for each configuration that corresponds to a boundary. Let the set of configurations that correspond to a boundary be represented by $\mathbf{C}_b$. Equation 17 is the first step in the derivation expanding out the likelihood into conditional probabilities.

$$P(W_O|\mathbf{C}_b) = \frac{P(W_O\mathbf{C}_b)}{P(\mathbf{C}_b)} \tag{17}$$

Since the real scene can not correspond to two different configurations I can expand equation 17 into equation 18.

$$P(W_O|\mathbf{C}_b) = \frac{\sum_{c\in\mathbf{C}_b} P(W_O \& c\in\mathbf{C}_b)}{\sum_{c\in\mathbf{C}_b} P(c\in\mathbf{C}_b)} \tag{18}$$

A slight change to equation 18 introduces the likelihoods generated by the algorithms in figures 8 through 13 and the prior probabilities that the scene is in a configuration tested by the algorithms. Equation 19 shows this change.

$$P(W_O|\mathbf{C}_b) = \frac{\sum_{c\in\mathbf{C}_b} P(W_O|c\in\mathbf{C}_b)P(c\in\mathbf{C}_b)}{\sum_{c\in\mathbf{C}_b} P(c\in\mathbf{C}_b)} \tag{19}$$

Equation 19 allows me (given priors on the templates or template sets) to gather many likelihoods into a single one. If I have likelihoods for every feature label and prior probabilities that the feature takes on that label I can use Bayes' law as in equation 3 (reprised here) to derive probabilities for feature labels given the data in the window.

$$P(l_f|O\&M)=\frac{P(O|l_f\&M)P(l_f|M)}{\sum_{l'_f\in L_f}P(O|l'_f\&M)P(l'_f|M)} \tag{3}$$

6.8. Estimating Boundaries

In section 6.7 I show how to derive probabilities given a likelihood generator. Often one must use programs (e. g. programs supplied as part of a package) that take as input estimates of the positions of the boundaries. Such programs can not use probabilities, they just want a boundary map. Here I show how to generate such an input.

To estimate where the boundaries are in an image it is necessary first to develop a cost function that describes what costs errors in estimation have. To use the probabilities of boundaries at points to estimate the configuration of boundaries in an image optimally it is necessary to use a cost function that sums the effects of pointwise errors. Such cost functions are simple to understand and require few parameters to describe (namely only the costs of different mislabelings at a point). I only use this type of cost function in this part of the paper. I also assume that making a correct decision has 0 cost.

For boundary point detection the costs that need to be calculated are:

(1) the cost of labeling a point as a boundary when there is no boundary there.

(2) the cost of labeling a point as not being a boundary when it is.

Call the cost of labeling a point (x,y) as a boundary point when it isn't $c_1(x,y)$ and the cost of labeling a point as not being a boundary when it is $c_2(x,y)$. Let $p_B(x,y)$ be the probability of a boundary at (x,y). Let $e_B(x,y)$ be 1 when the estimation procedure indicates there is a boundary at (x,y) and 0 otherwise. We want a detector that minimizes the expected cost for the estimation. Thus we want to minimize the summation in equation 20.

$$\sum_{(x,y)} c_1(x,y)(1-p_B(x,y))e_B(x,y)+c_2(x,y)p_B(x,y)(1-e_B(x,y)) \tag{20}$$

Let us assume that $c_1(x,y)$ is the same for all (x,y) and the same for c_2. Equation 20 is clearly minimized by minimizing equation 21 for each (x,y) ((x,y) is deleted for clarity).

$$c_1(1-p_B)e_B+c_2p_B(1-e_B) \tag{21}$$

Equation 21 can be rearranged into equation 22.

$$(c_1(1-p_B)-c_2p_B)e_B(x,y)+c_2p_B \tag{22}$$

Clearly you want e_B to be 1 when equation 23 is positive and e_B to be 0 when equation 23 is negative.

$$c_1(1-p_B)-c_2p_B \tag{23}$$

This statement can be algebraically transformed into the statement that e_B should be 1 when the inequality in equation 24 is satisfied and 0 otherwise.

$$p_B < \frac{c_1 - c_2}{c_1} \tag{24}$$

Thus one only needs to threshold the probabilities of boundaries with $\frac{c_1 - c_2}{c_1}$ to estimate the positions of the boundary for the additive cost function with costs c_1 and c_2. This argument is standard in Bayesian decision theory with simple loss functions [Berger80].

7. Implementation Details

Here, I describe my implementation of the algorithms described in section 6. I have code for the algorithms in figures 8, and 10. I also have constructed the code that is implied by equations 19 and 3 of section 6.7.

7.1. Likelihood Generators

These algorithms are based on the assumption that the scene can be modeled by a set of templates. The templates are objects of unit reflectance under unit lighting. I have one template that represents the window being in the interior of the object shown in figure 16.

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Figure 16: Template for the Interior of an Object

For each of 4 directions, 0 degrees, 45 degrees, 90 degrees, and 135 degrees, I have 3 pairs of templates that describe three possible boundaries. For example figure 17 shows the 0 degree templates.

1^{st} pair

1	1	0.5	0	0
1	1	0.5	0	0
1	1	0.5	0	0
1	1	0.5	0	0
1	1	0.5	0	0

0	0	0.5	1	1
0	0	0.5	1	1
0	0	0.5	1	1
0	0	0.5	1	1
0	0	0.5	1	1

2^{nd} pair

1	1	0	0	0
1	1	0	0	0
1	1	0	0	0
1	1	0	0	0
1	1	0	0	0

0	0	1	1	1
0	0	1	1	1
0	0	1	1	1
0	0	1	1	1
0	0	1	1	1

3^{rd} pair

1	1	1	0	0
1	1	1	0	0
1	1	1	0	0
1	1	1	0	0
1	1	1	0	0

0	0	0	1	1
0	0	0	1	1
0	0	0	1	1
0	0	0	1	1
0	0	0	1	1

Figure 17: Template for the 0 degree Boundary

Each pair of templates represent two objects, one occluding the other. I have not generated any templates for windows with 3 objects.

The algorithms in figures 8, and 10 consist of a part that is dependent on a template used and a part that is a function of the observed window. It is the part that depends on the template that presents implementation difficulties. Function F_2 is a function of the sum of squared elements in the template. Function F_4 is a function of the pairwise products of the templates representing the objects in the configuration. Both of these functions were implemented by table lookup with linear

interpolation in my implementation For every direction the sum of squares of the templates and the pairwise products are described by the same 5 numbers. Only two F_4 tables need to be generated.

F_2 and F_4 also depend on the standard deviation of the noise in the image. I call a likelihood generator that assumes a specified standard deviation of noise a likelihood generator *tuned* to that standard deviation of noise. I have likelihood generators tuned to noise with σ equal to 4, 8, 12, and 16.

7.2. Probabilities from Likelihoods

I use equation 19 to gather together the three likelihoods generated from the 3 pairs of templates in each direction into a single likelihood. Thus I have the likelihoods for the four directions that a boundary passes through or next to the center pixel.

I also need the likelihood that there is no boundary near or through the center pixel. To get this likelihood I take the likelihood that the window is in the interior of the object and combine it with likelihoods for central boundaries calculated for the neighboring windows. Thus from the 0 degree boundary likelihoods I use the likelihood from the window one pixel left and right and combine it with the likelihood of a noncentral edge.

Thus I have 4 likelihoods for 4 directed boundary points and one likelihood that represents the likelihood that there is no central edge in the image. I then use Bayes' law from equation 3 to compute the probabilities of these 4 states. I then threshold the probabilities at 0.5 to present the results shown in section 8. Throughout I assumed that the prior probability of a central edge is 0.1 and that this probability is equally distributed in all 4 directions. This prior information is sufficient to apply Bayes' law.

8. Results from Implemented Boundary Detectors

I have implemented the algorithms in sections 6.1 and 6.3 figures 8 and 10. Here I describe the results from testing this detector.

The software I have written is flexible. However I have only constructed templates for a restricted set of configurations. I have templates for a step edge model with the low curvature assumption, 4 possible orientations for boundaries, and 255 gray levels in the image. My templates handle boundaries that occur in the center of pixels or between pixels. I have built the templates for a 5x5, 7x7 and 9x9 windows. I also have constructed tables to compute F_1 and F_4 for each of these windows that assume noise of standard deviations 4,8,12, and 16.

8.1. Results with Artificial Images

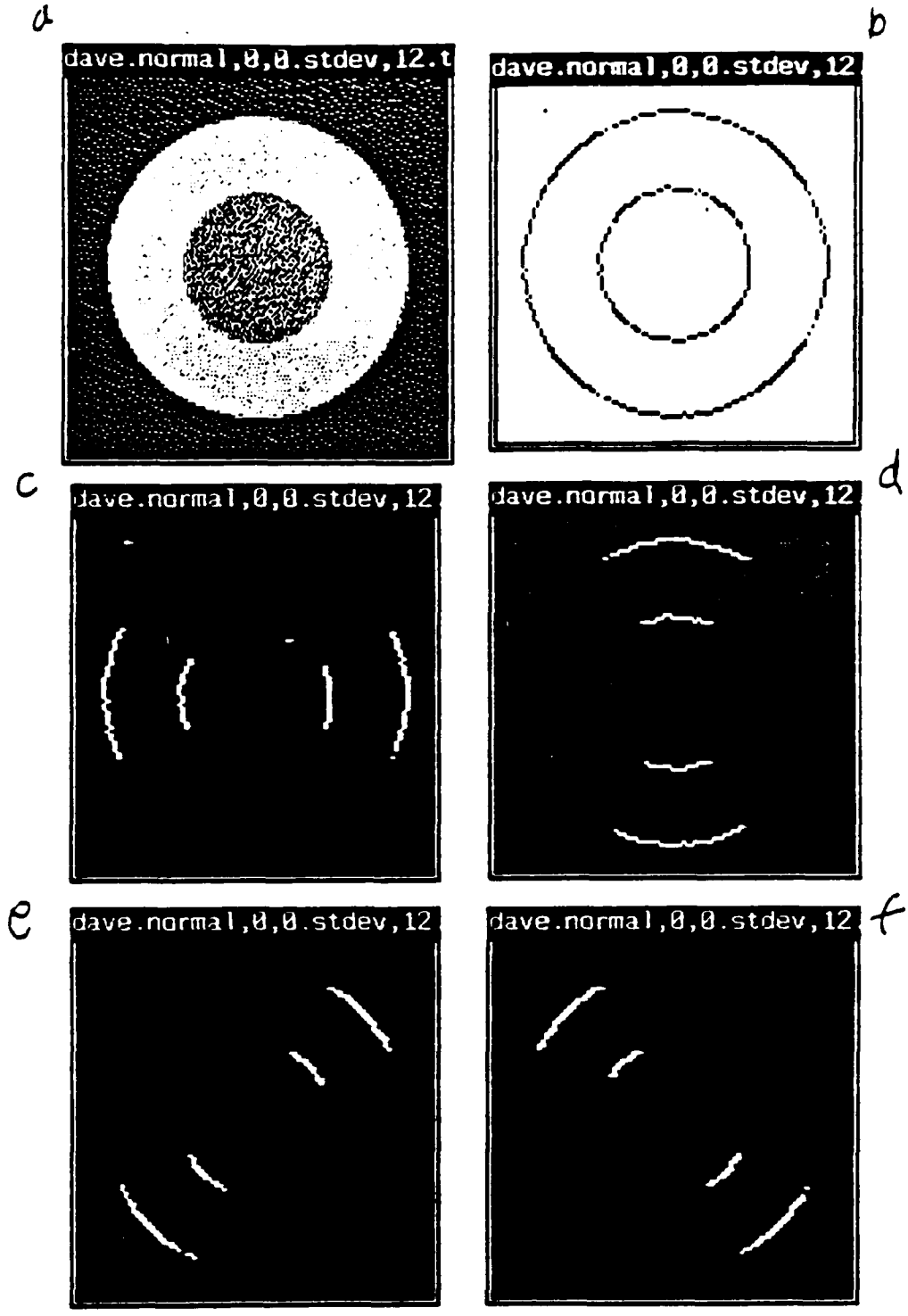
I have applied these operators to test images constructed by a package of graphics routines. This package was written by Myra Van Inwegen and is described in an upcoming technical report.

I describe my operators applied to two test images generated by this package One is an image of two circles shown on the left in figure 18a

Figure 18: Artificial Images

A more challenging and complex image has also been tested shown in figure 18b.

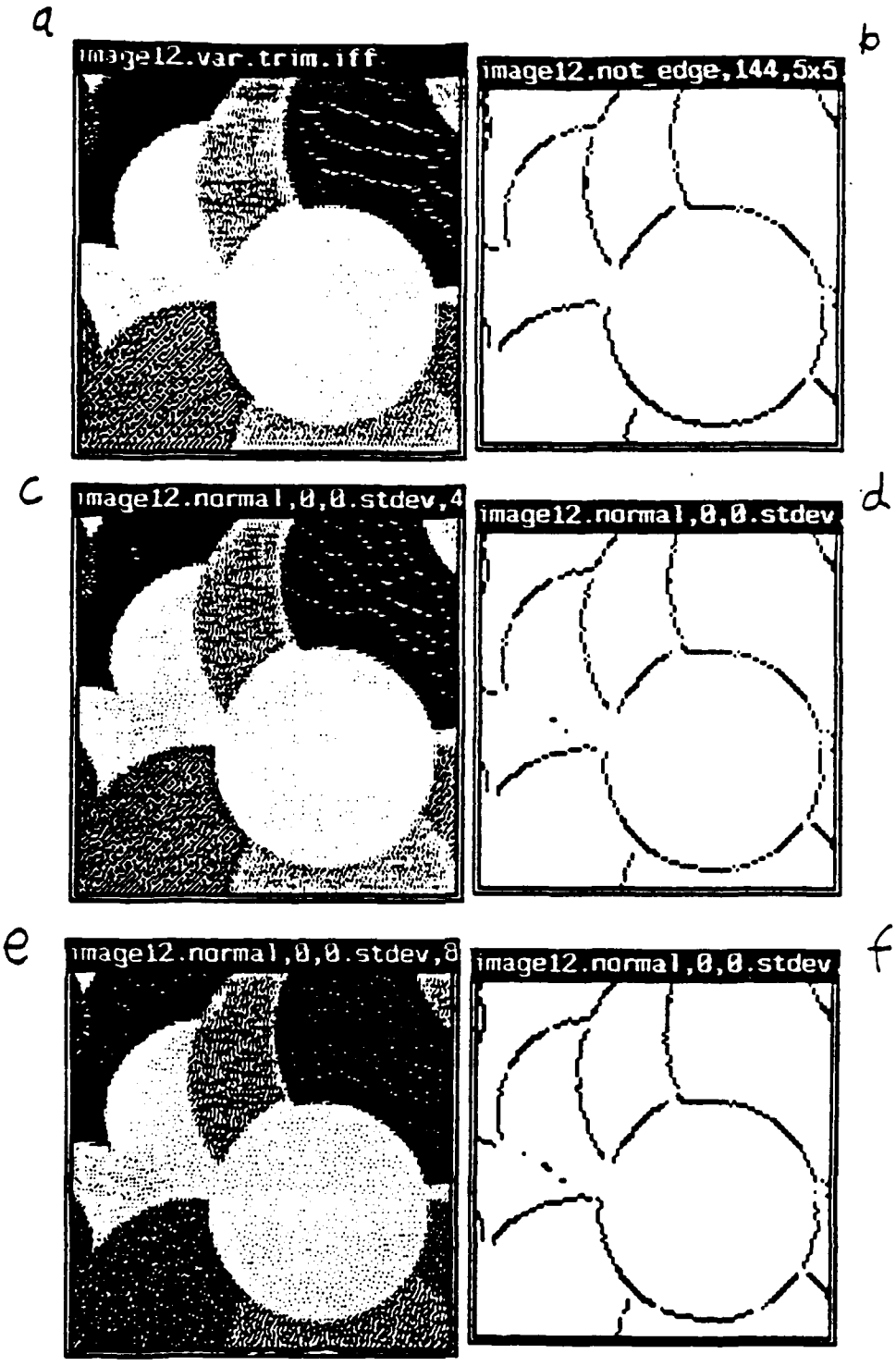
The two circle image (figure 18a) is a particularly good image to test the effect of boundary orientation, curvature and contrast on boundary detection. Figure 19 shows the result of using a 5x5 operator tuned to standard-deviation 12 noise on image 18a with standard deviation 12 noise added to it. The images are white at points of greater than 50% probability and black at points less than 50% probability.



a: image with stdev 12 noise
b: white if there is no edge
c: white if 0 degree edge
d: white if 90 degree edge
e: white if 45 degree edge
f: white if 135 degree edge

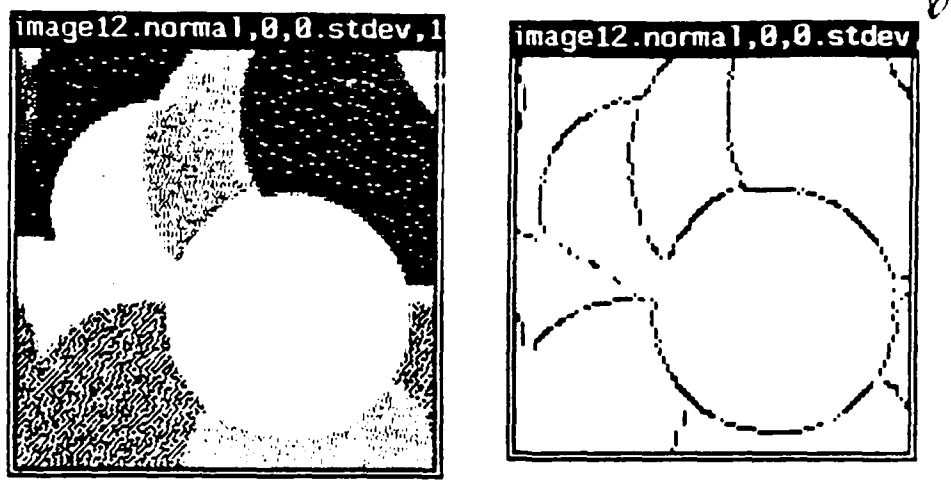
Figure 19: Oriented response for 5x5 operator

In figures 20, 21, and 22 I apply the standard deviation 12 operator to the image 18b with too little, just right and too much noise respectively. The operator output is black when there is greater than 50% probability of a boundary. Note that with too little noise the detector misses boundaries that are there. With too much noise the detector detects boundaries that are not there.



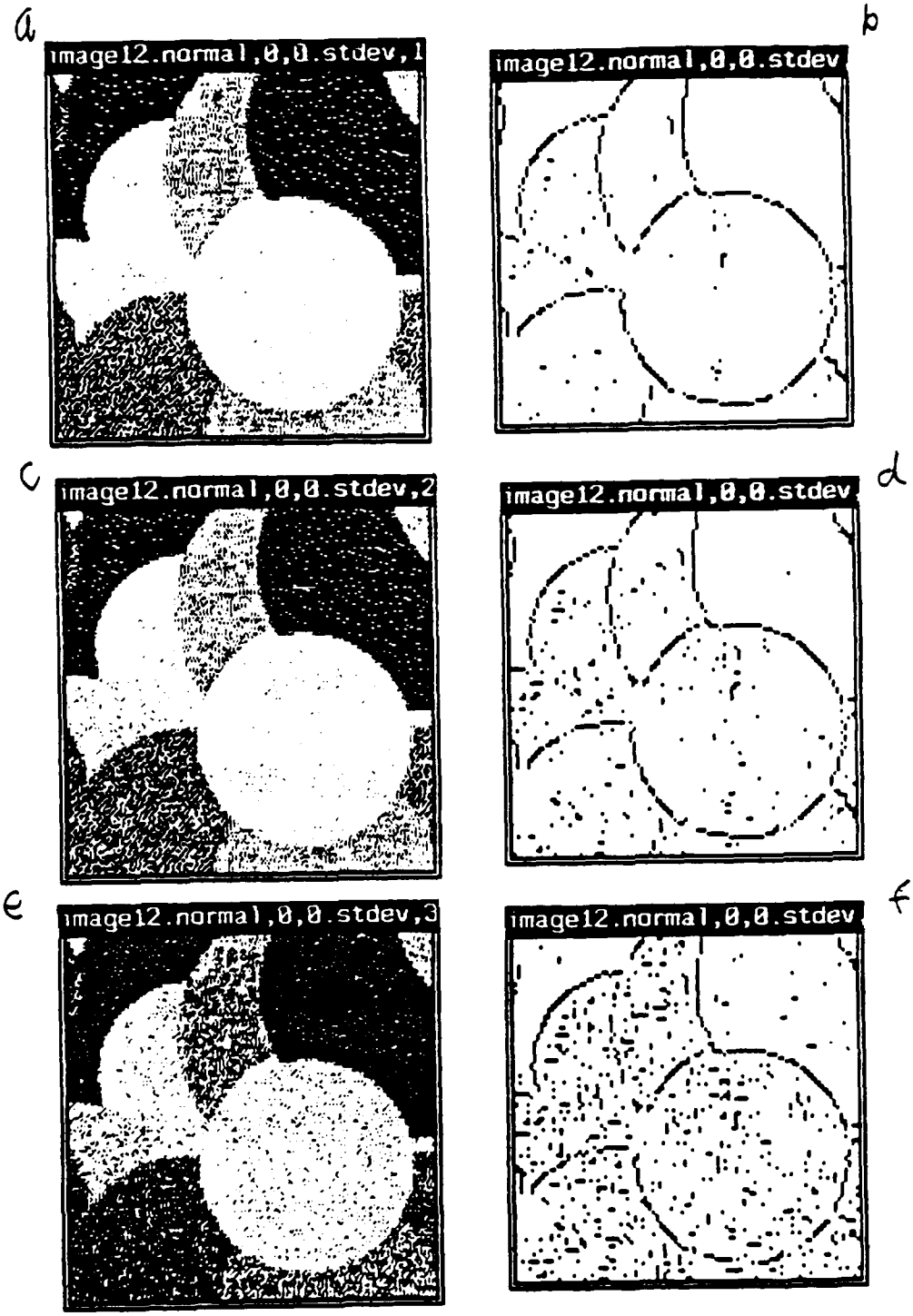
a: image with $\sigma=0$ noise
b: $\sigma=12$ operator on image with $\sigma=0$ noise
c: image with $\sigma=4$ noise
d: $\sigma=12$ operator on image with $\sigma=4$ noise
e: image with $\sigma=8$ noise
f: $\sigma=12$ operator on image with $\sigma=3$ noise

Figure 20: $\sigma=12$ 5x5 operator applied to images with too little ($\sigma<12$) noise



a: image with $\sigma = 12$ noise b: $\sigma = 12$ operator on image with $\sigma = 12$ noise

Figure 21: $\sigma = 12$ 5x5 operator applied to image with correct ($\sigma = 12$) amount of noise



a: image with $\sigma=16$ noise b: $\sigma=12$ operator on image with $\sigma=16$ noise
c: image with $\sigma=20$ noise d: $\sigma=12$ operator on image with $\sigma=20$ noise
e: image with $\sigma=32$ noise f: $\sigma=12$ operator on image with $\sigma=32$ noise

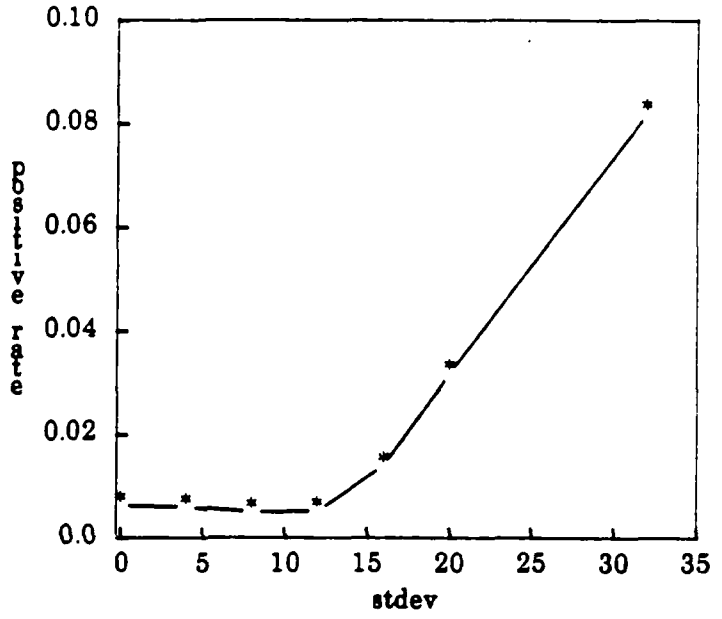
Figure 22: $\sigma=12$ 5x5 operator applied to images with too much ($\sigma>12$) noise

Because these are artificial images I know exactly where the boundaries are. Thus I have written software that counts how many mistakes (false positives and negatives) are made in boundary determination. False negatives are when a boundary that is in the image is missed. False positives are boundaries that are reported where there is no boundary there.

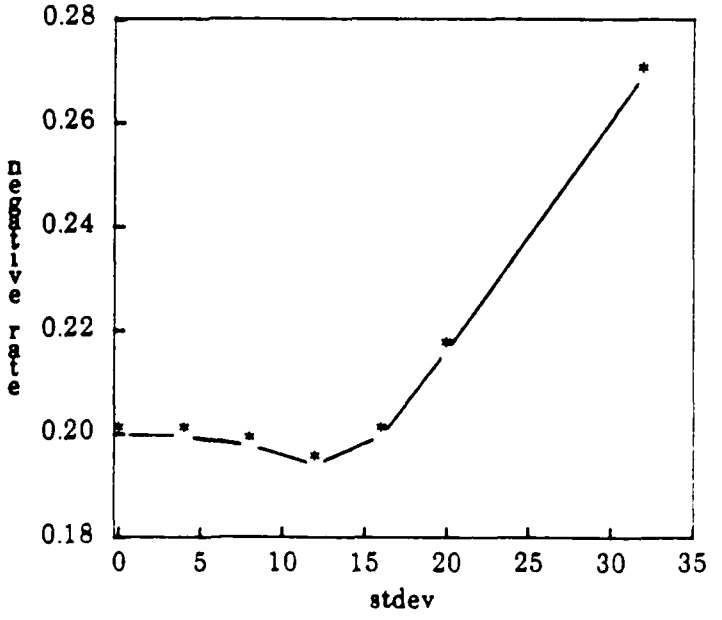
One tricky point is that multiple reports of boundaries are usually considered a bad result [Canny83]. However systems that report a boundary only once will usually have a high false negative rate because they report an edge one pixel off from where it really is. I consider this error to have low enough cost to be ignored. So my software ignores false negatives that are next to reported boundaries in a direction normal to the boundary.

Figure 23 charts the performance of my operator tuned to $\sigma=12$ on the images shown previously. Figure 24 charts the performance of my operator tuned to $\sigma=4$ noise. Figure 25 charts the performance of an operator that is tuned to the same noise level as is contained in the image. Figure 26 superimposes the three graphs of total error rates to show the relationship.

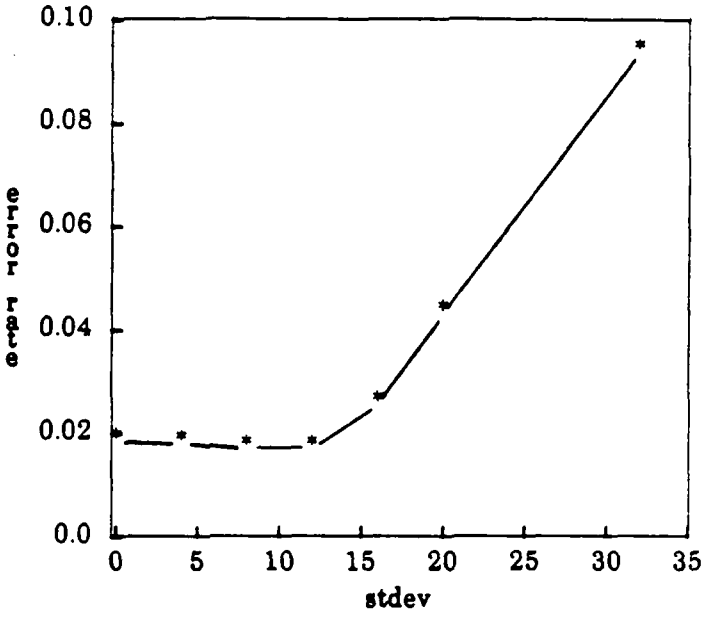
(a)



(b)

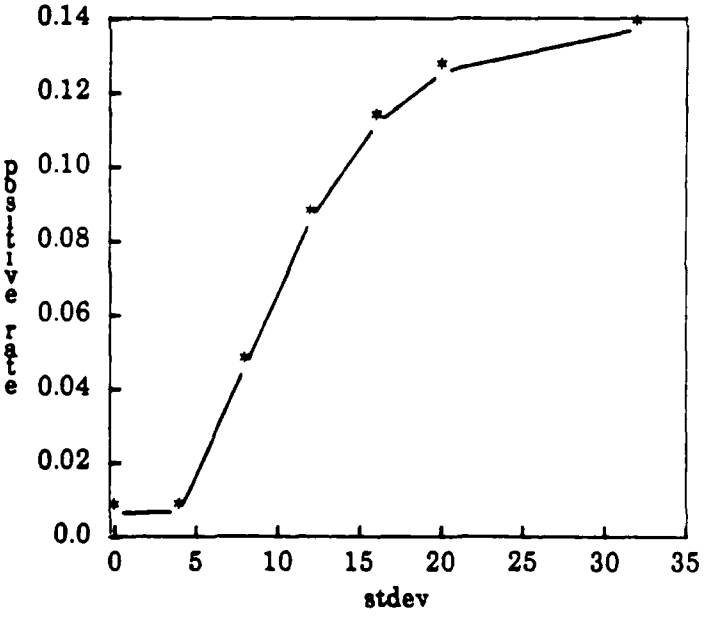


(c)

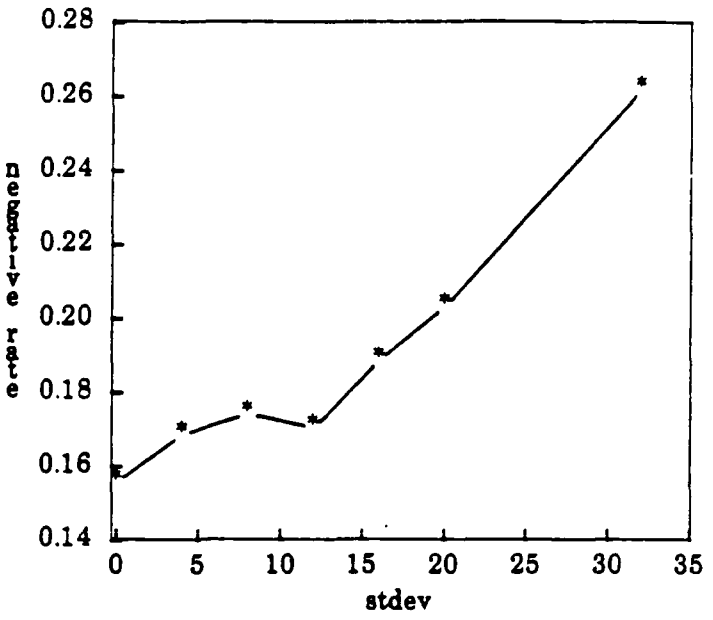


(a) false positive rate vs increasing σ of noise in image
(b) false negative rate vs increasing σ of noise in image
(c) total error rate vs increasing σ of noise in image
Figure 23: Error Rates for the Operator Tuned to $\sigma = 12$ noise

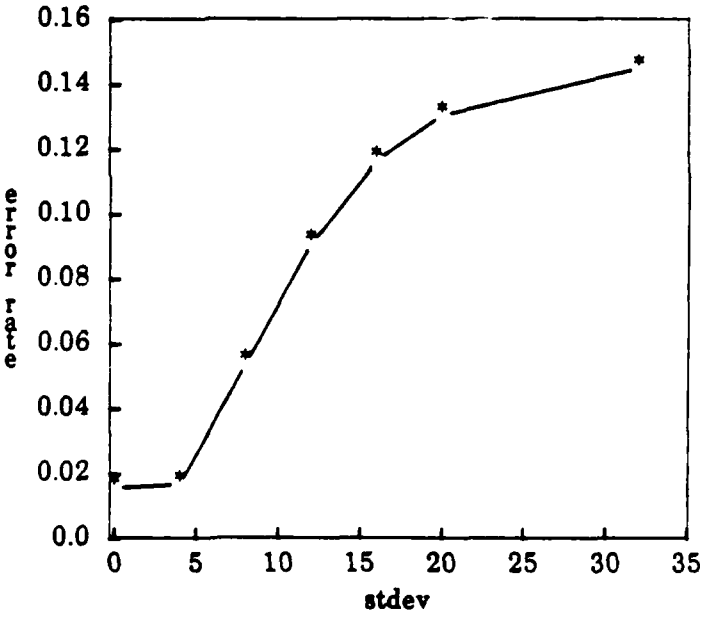
(a)



(b)

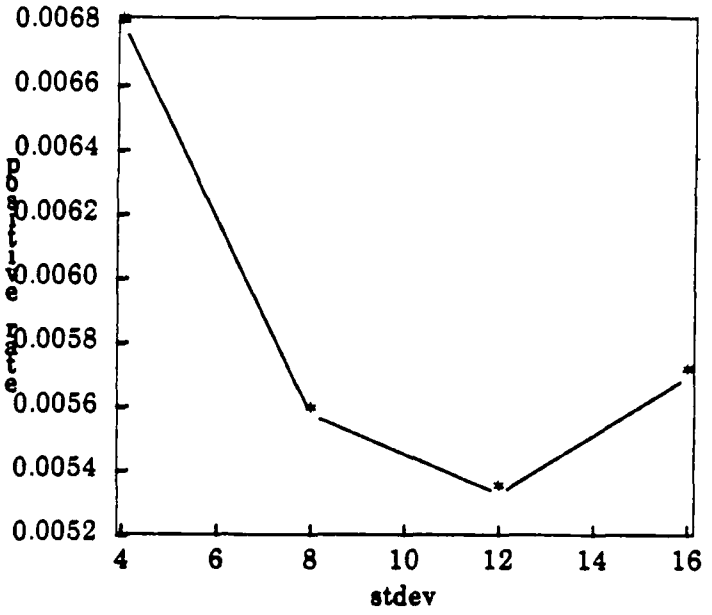


(c)

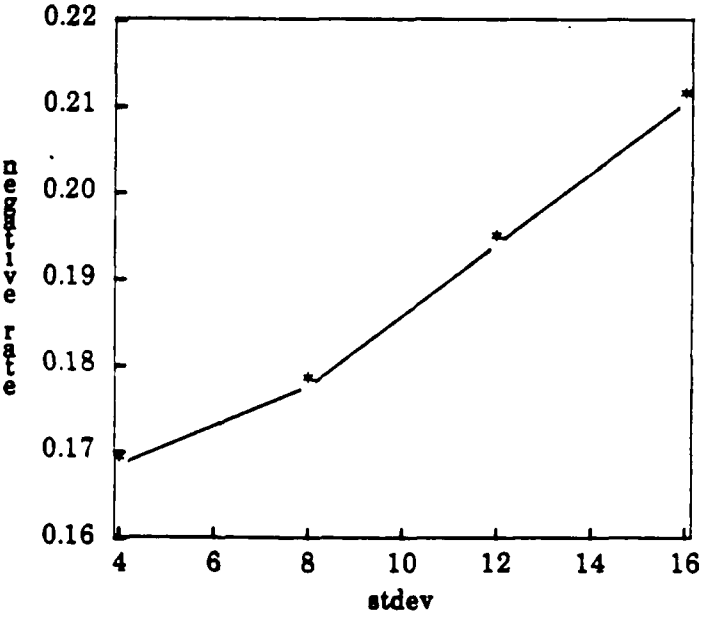


(a) false positive rate vs increasing σ of noise in image
(b) false negative rate vs increasing σ of noise in image
(c) total error rate vs increasing σ of noise in image
Figure 24: Error Rates for the Operator Tuned to $\sigma=4$ noise

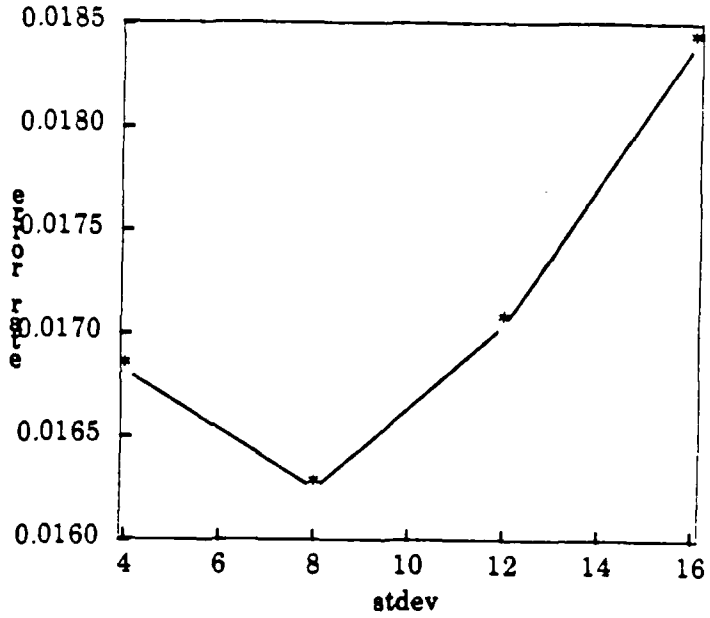
(a)


0.0068
0.0066
0.0064
0.0062
0.0060
0.0058
0.0056
0.0054
0.0052
4
6
8
10
12
14
16
stdev

(b)

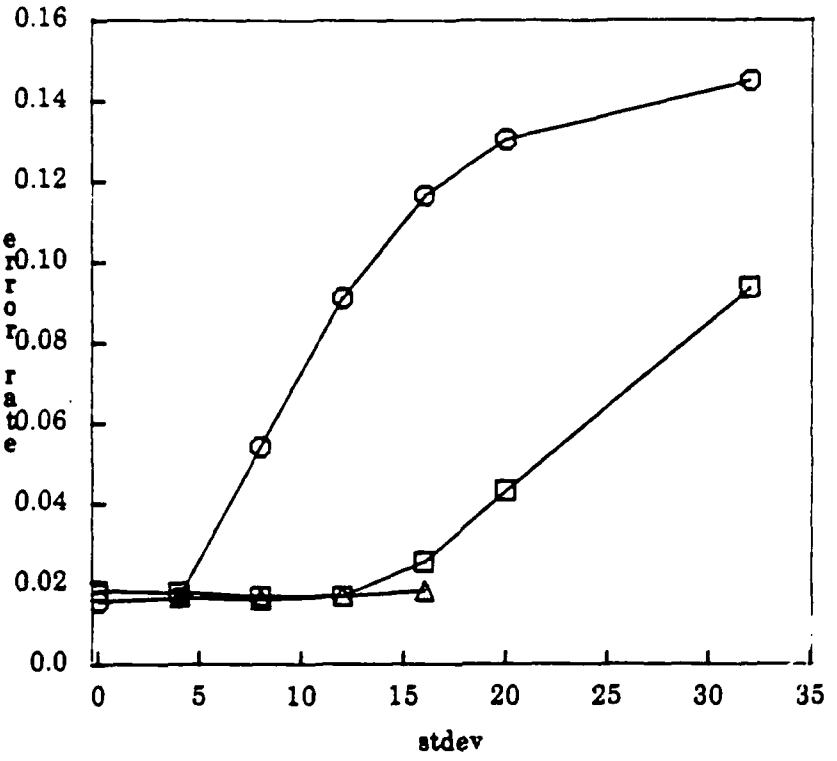

0.22
0.21
0.20
0.19
0.18
0.17
0.16
4
6
8
10
12
14
16
stdev

(c)



(a) false positive rate vs increasing σ of noise in image
(b) false negative rate vs increasing σ of noise in image
(c) total error rate vs increasing σ of noise in image

Figure 25: Error Rates for the Operator Tuned to σ of the noise in the Image



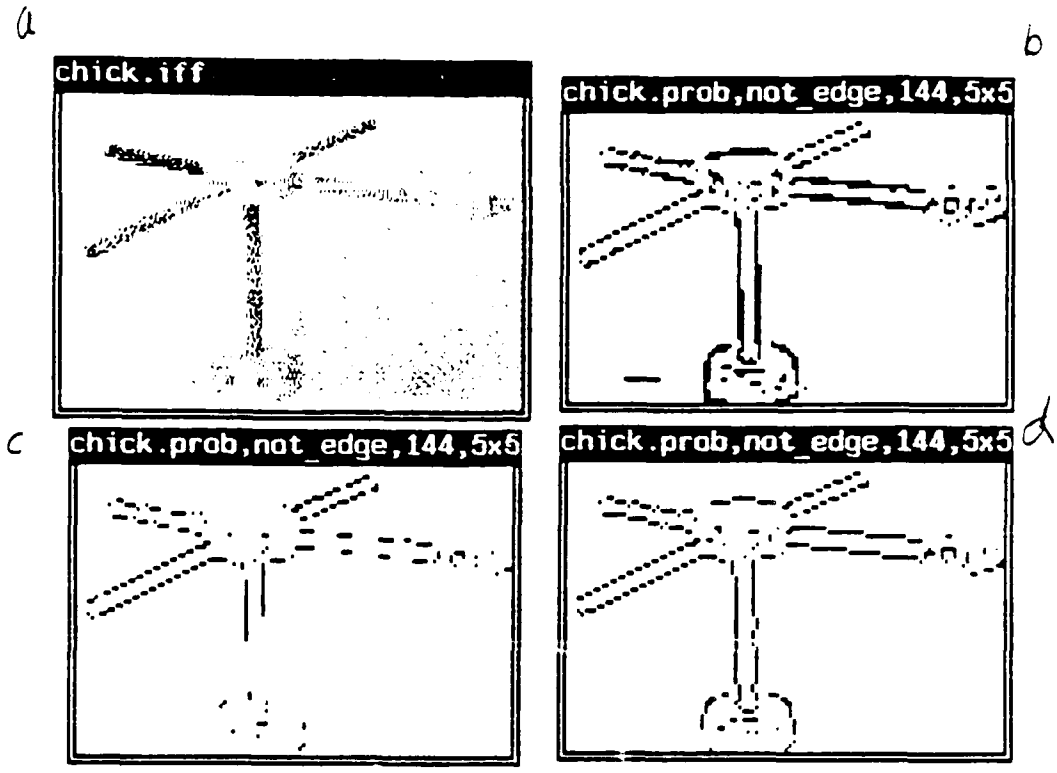
circles: Operator Tuned to $\sigma = 12$
squares: Operator Tuned to $\sigma = 4$
triangles: Operator Tuned to noise in image

Figure 26: Total error rate for my operators

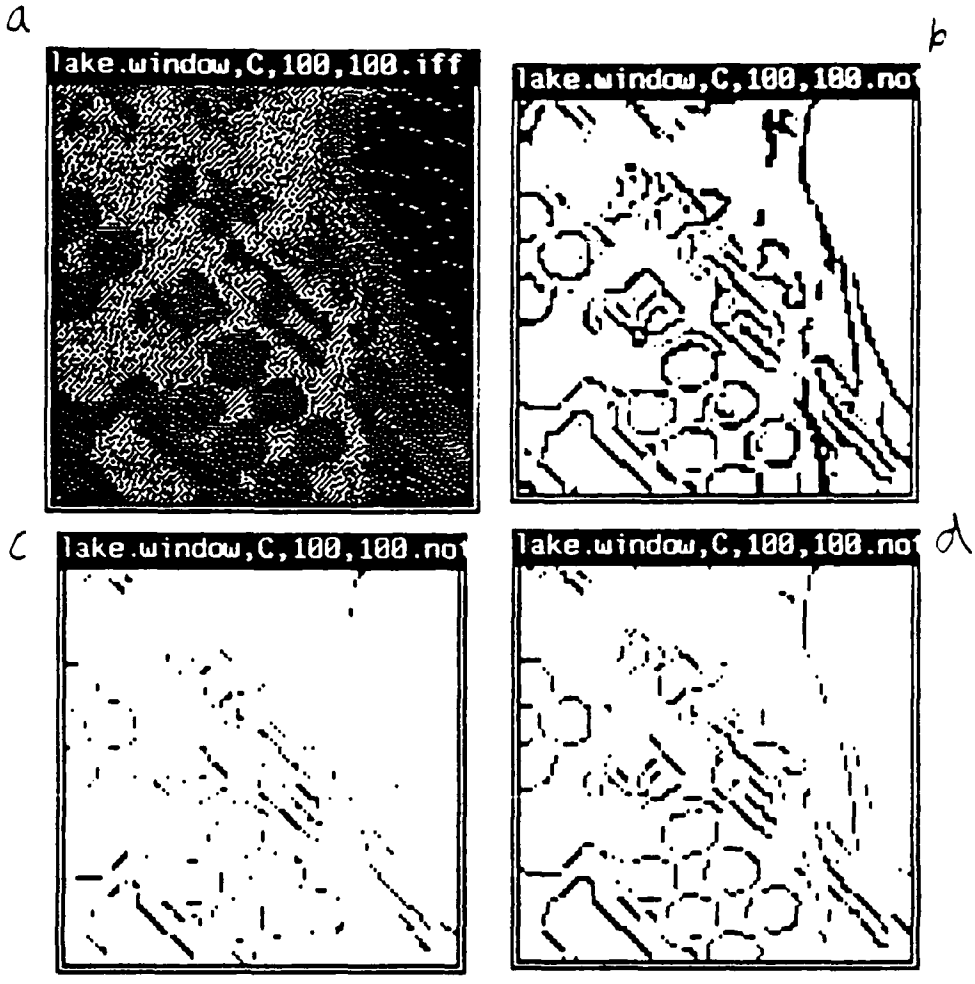
As you can see the tuned operator is always at least as good as the operator that has been developed for stdev 12 or 4 noise.

8.2. Results with Real Images

I have also applied my operator to real images. Here, I use two images, a laboratory image of a Tinkertoy model (figure 27) and an aerial picture (figure 28). I demonstrate the utility of having operators that return probabilities with these results. In both these figures (a) is the image, (b) is the output of my 5x5 stdev 12 operator thresholded at 10% probability, (c) is thresholded at 90% probability and (d) is thresholded at 50% probability. (b) would be used when the cost of missing an edge is high as when the output would be fed to a regularization technique. Note that for case (b) the operator sometimes returns thickened edges. (c) would be used when the cost of missing an edge is low. Hough transform techniques are often developed with that assumption in mind. There is enough information in figure 27c to find the rods of the tinker toy even though all the boundaries are not extant. (d) is what you use when an operator is equally troubled by all errors.



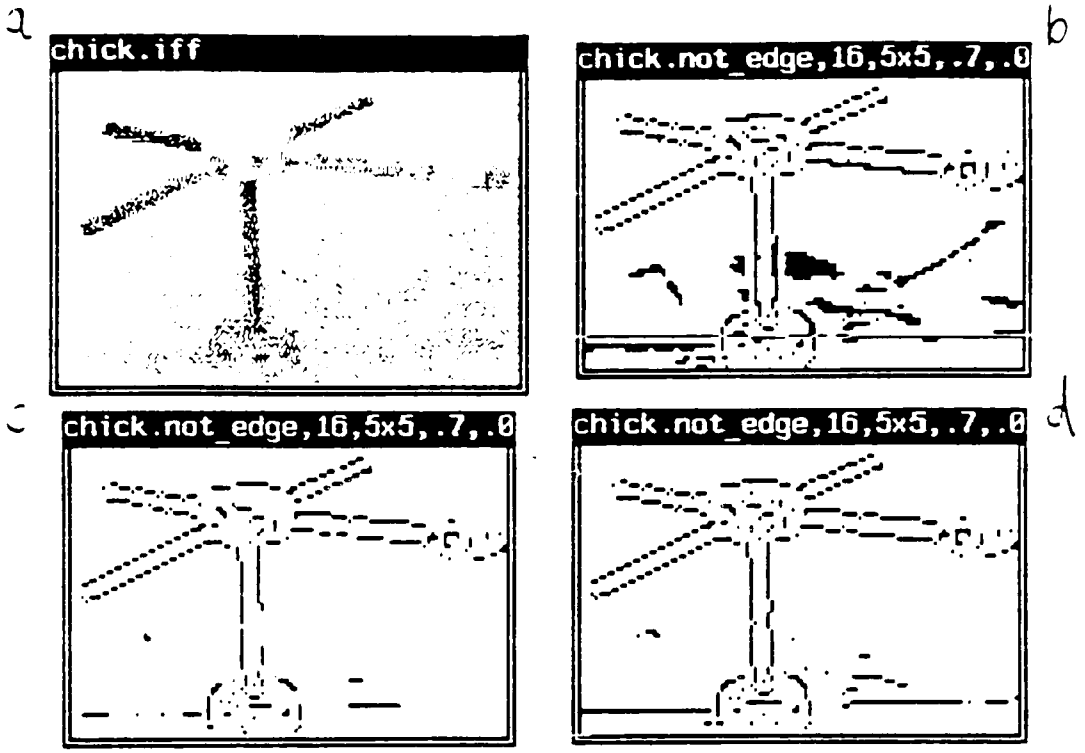
(a) Tinkertoy Image
(b) Output of $\sigma = 12$ Operator with threshold at .1
(c) Output of $\sigma = 12$ Operator with threshold at .9
(d) Output of $\sigma = 12$ Operator with threshold at .5
Figure 27: $\sigma = 12$ 5x5 operator applied to tinker toy image



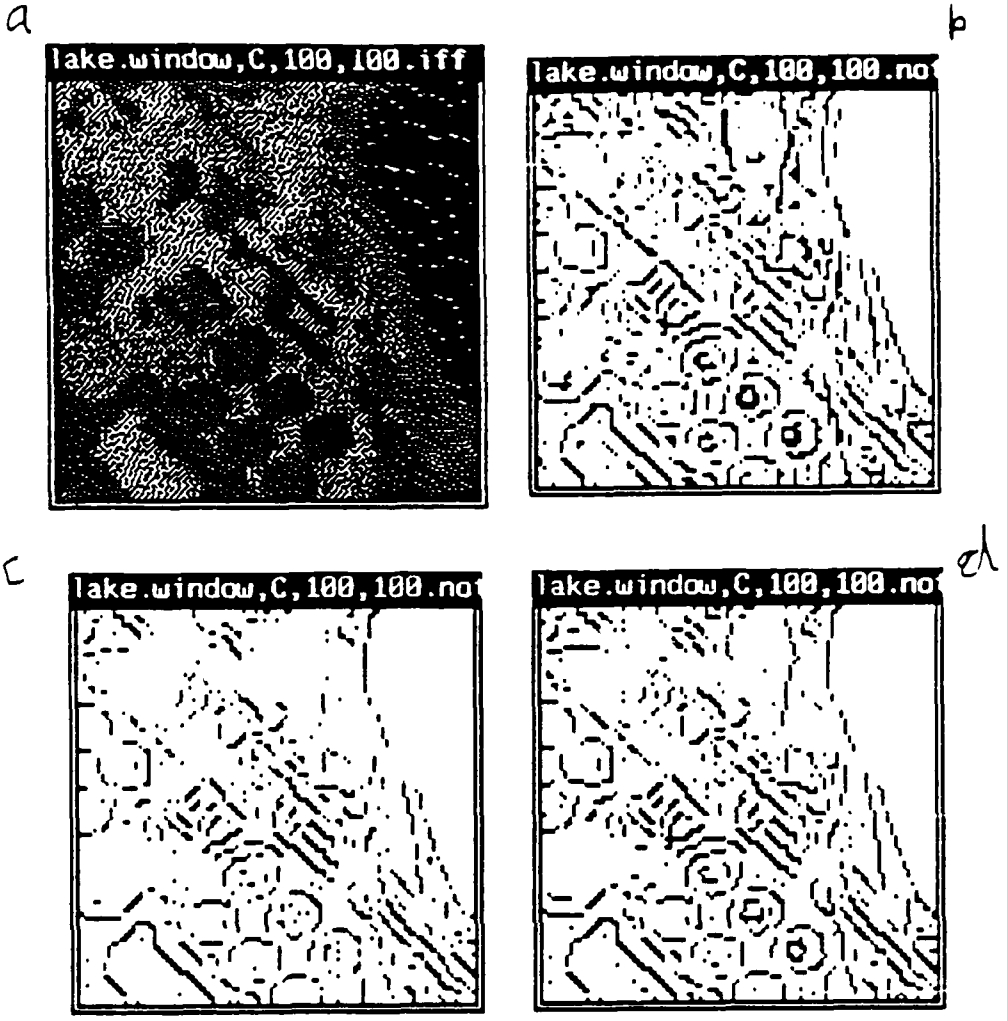
(a) Aerial Image
(b) Output of $\sigma = 12$ Operator with threshold at .1
(c) Output of $\sigma = 12$ Operator with threshold at .9
(d) Output of $\sigma = 12$ Operator with threshold at .5
Figure 28: $\sigma = 12$ 5x5 operator applied to aerial image

A particular threshold may result in a most pleasing (to a human observer) ensemble of results (perhaps .5). But this threshold may not be the best threshold for the succeeding application.

One may notice that lowering the threshold seems to increase the number of boundary points in the regions of real boundaries. It is not surprising that the regions that look most like boundaries should be near boundaries. Also the probabilities of boundaries are being reported as lower than they should be in this image. A good reason for this is that the model used to construct the operator shown is not a good model for this image. In particular there is some evidence [Sher87] that the standard deviation of the noise is closer to 4 than 12 in these images. Thus look at the same results for the $\sigma = 4$ operator in figures 29 and 30.



(a) Tinkertoy Image
(b) Output of $\sigma = 4$ Operator with threshold at .1
(c) Output of $\sigma = 4$ Operator with threshold at .9
(d) Output of $\sigma = 4$ Operator with threshold at .5
Figure 27: $\sigma = 4$ 5x5 operator applied to tinker toy image



(a) Aerial Image
(b) Output of $\sigma = 4$ Operator with threshold at .1
(c) Output of $\sigma = 4$ Operator with threshold at .9
(d) Output of $\sigma = 4$ Operator with threshold at .5
Figure 28: $\sigma = 4$ 5x5 operator applied to aerial image

8.3. Comparisons with Established Techniques

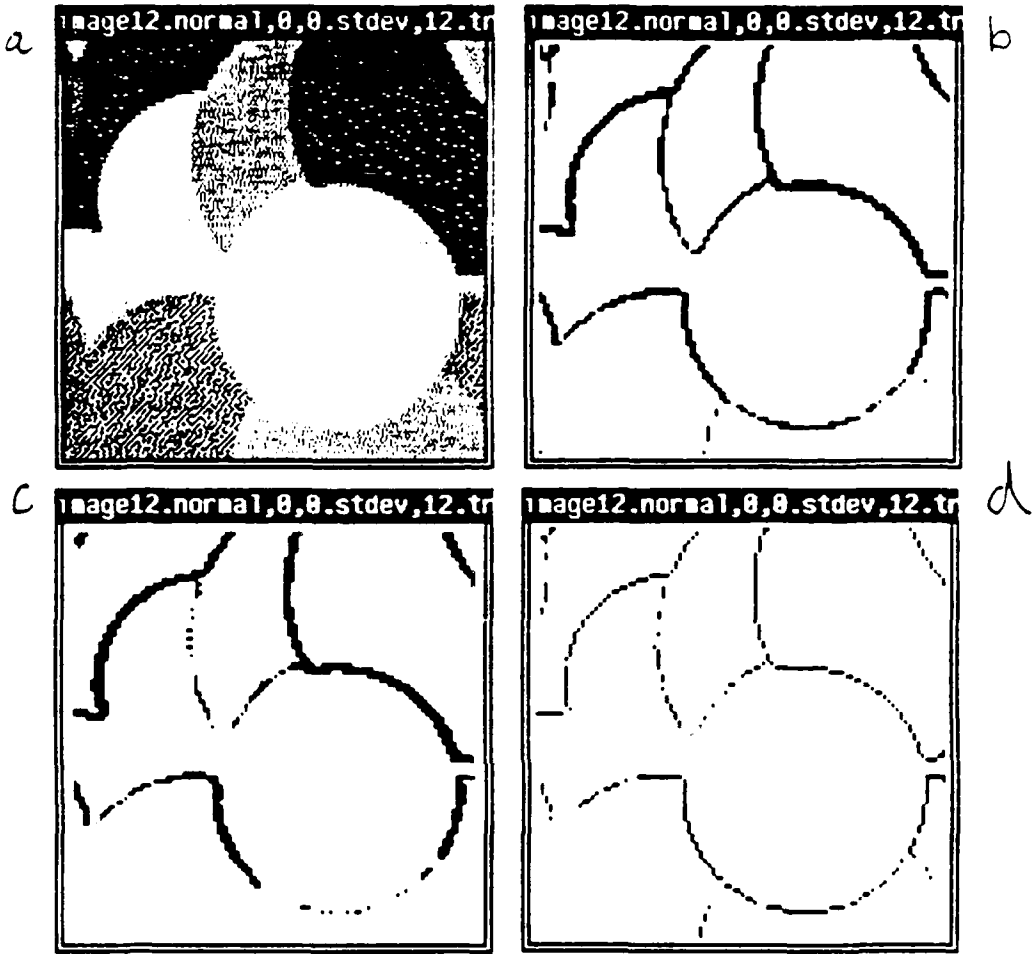
Here, I compare the results I have just presented with established edge detectors. The edge detectors I currently have results for are the Sobel thresholded at 200, the 5 by 5 Kirsch thresholded at 750, and a thinned 3 by 3 Kirsch thresholded at 100. The thresholds for the Sobel and the thinned Kirsch were found to be the ones that minimized the number of errors when applied to the artificial image in figure 18b with standard deviation 12 noise added. The threshold that minimized the errors with the Kirsch was 1175. However at this threshold more than 50% of the boundaries were missed. It was difficult to chart the result of using this operator along with the rest[2] so I used a somewhat lower threshold.

I know these operators are not state of the art. However these results show that tools are available to test this theory against more sophisticated operators. More sophisticated operators

[2]Visual inspection indicated that lowering the threshold would make the result of the operator more appealing. Not that it should matter but

such as Canny's and Haralick's will be tested against my operators in the next month or two.

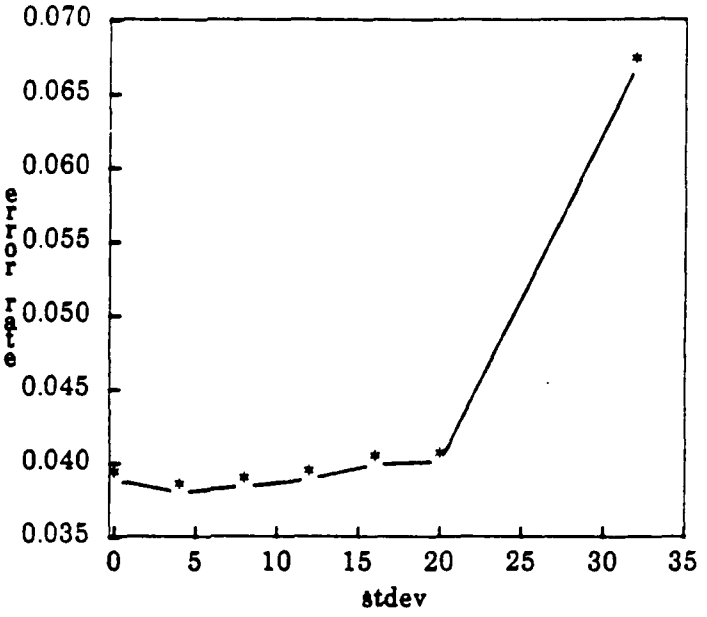
In figure 31 the results of applying the Sobel, Kirsch and thinned Kirsch to my artificial image with stdev 12 noise.



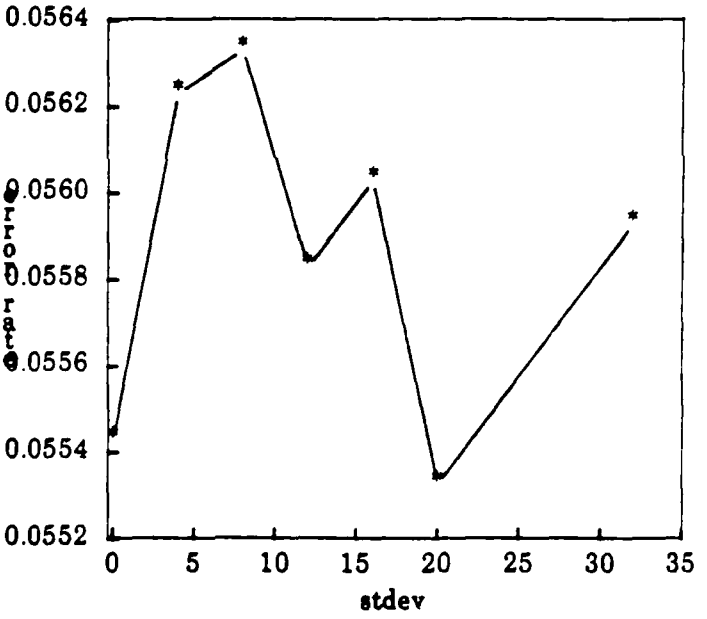
(a) Image with $\sigma = 12$ noise.
(b) Output of the Sobel Optimally Thresholded
(c) Output of the 5x5 Kirsch Optimally Thresholded
(d) Output of the Thinned Kirsch Optimally Thresholded
Figure 31: Application of Established Operators to an Artificial Image

I have measured the error rates for these operators and have charted the total error rates in figure 32.

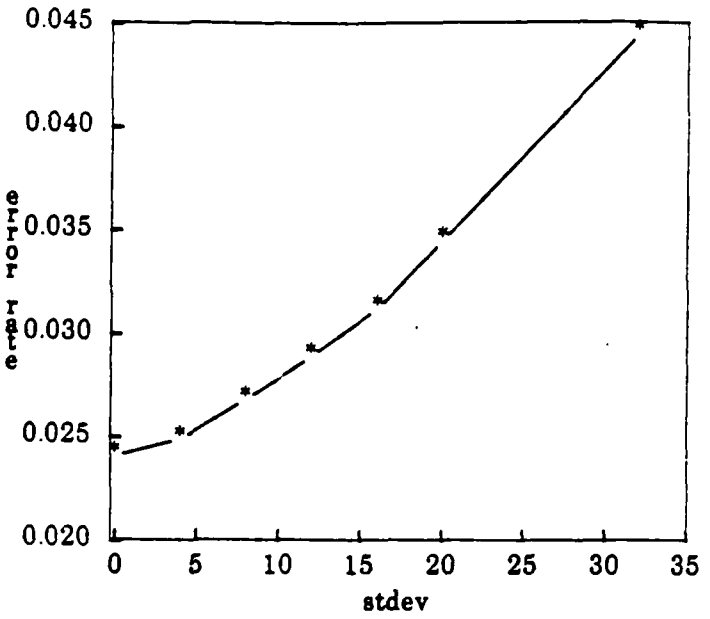
(a)


0.070
0.065
0.060
0.055
0.050
0.045
0.040
0.035
error rate
0
5
10
15
20
25
30
35
stdev

(b)


0.0564
0.0562
0.0560
0.0558
0.0556
0.0554
0.0552
error rate
0
5
10
15
20
25
30
35
stdev

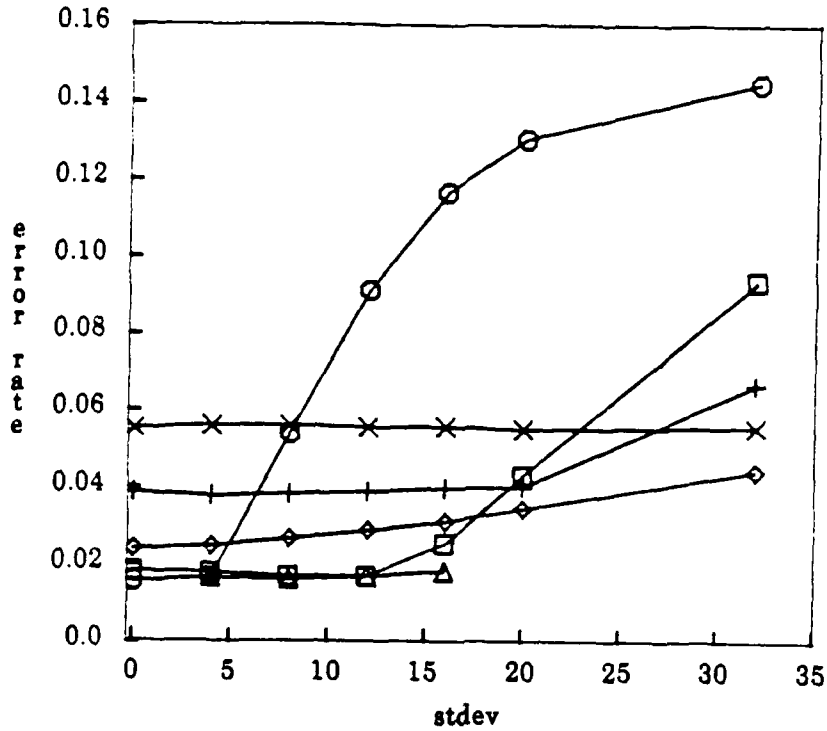
(c)



(a) Error rates for the thresholded Sobel
(b) Error rates for the thresholded Kirsch
(c) Error rates for the thinned Kirsch

Figure 32: Charts of Total Error Rates for Established Operators

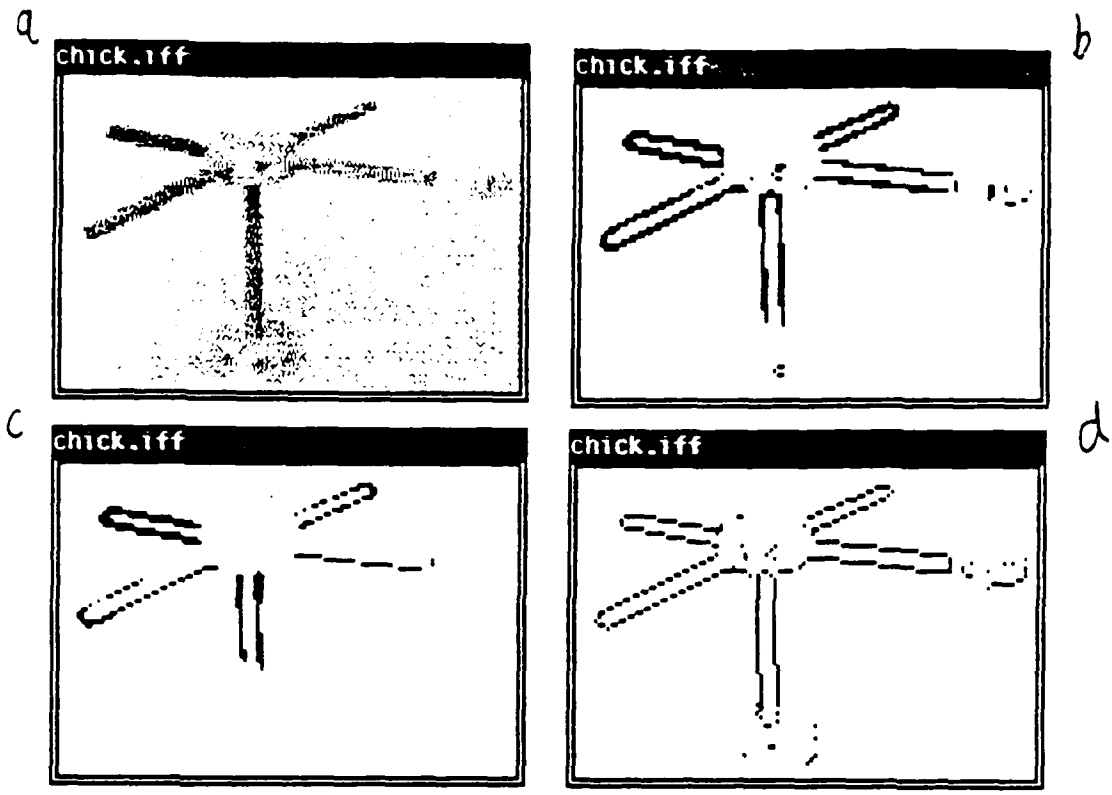
To summarize the results I have a plot that shows the error rates for all the operators I have tested so far (figure 33). In this chart my stdev 12 operator is shown as squares, my tuned operator is shown as circles, the Sobel is shown as triangles, the thresholded Kirsch is shown as crosses and the thinned Kirsch is shown as X's.



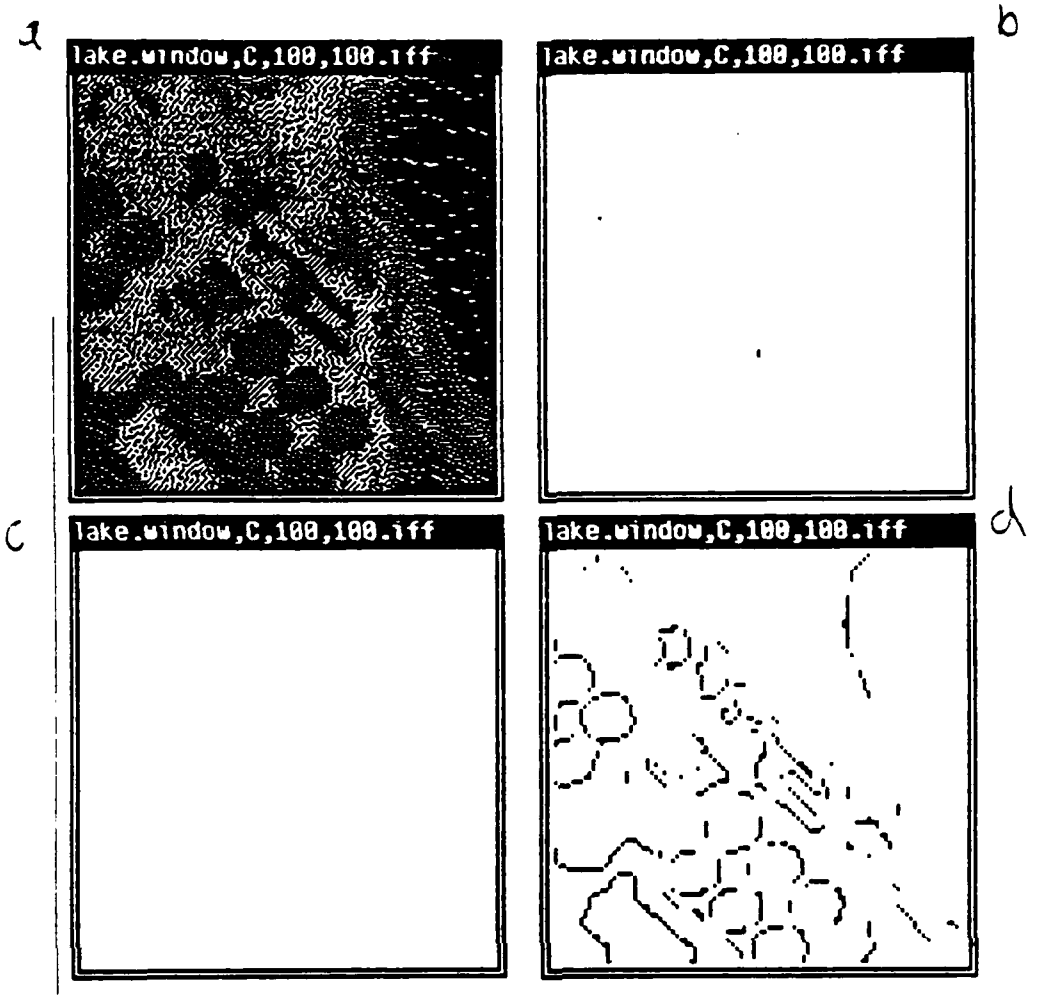
circles:	Tuned $\sigma = 12$ operator
squares:	Tuned $\sigma = 4$ operator
triangles:	Tuned to noise σ operator
crosses:	Sobel's error rate
X's:	5x5 Kirsch's error rate
diamonds:	Thinned Kirsch's error rate

Figure 33: Error Rates for all Operators

For comparison, I have run the 3 established edge detector on the two real images shown in section 8.2. Figures 34 and 35 show the result of runing these operators with my established operators. Clearly, in these circumstances the most effective established operator for these images is the thinned Kirsch.



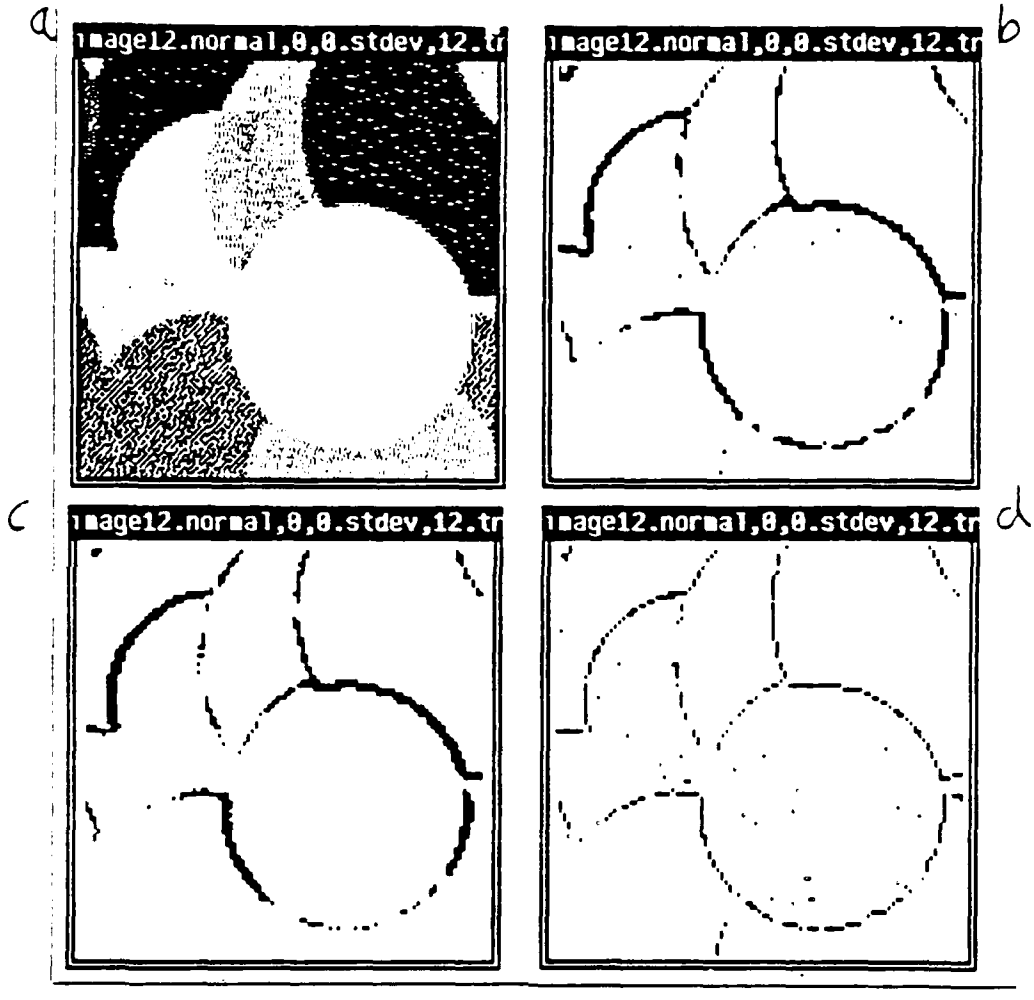
(a) Tinkertoy Image
(b) Output of the Sobel Optimally Thresholded
(c) Output of the 5x5 Kirsch Optimally Thresholded
(d) Output of the Thinned Kirsch Optimally Thresholded
Figure 34: Application of Established Operators to Tinkertoy Image



(a) Aerial Image
(b) Output of the Sobel Optimally Thresholded
(c) Output of the 5x5 Kirsch Optimally Thresholded
(d) Output of the Thinned Kirsch Optimally Thresholded
Figure 35: Application of Established Operators to Aerial Image

One can criticize these comparisons by saying that the image statistics favor my operators, which are robust with dim images. To counter this criticism I have also run the three operators (Sobel, Kirsch and thinned Kirsch) with a preprocessing stage of histogram equalization. Thus all the test images will be rescaled to have the same statistics. Thus when I find the optimal threshold for the standard deviation 12 artificial image it should remain a good threshold for all the tests.

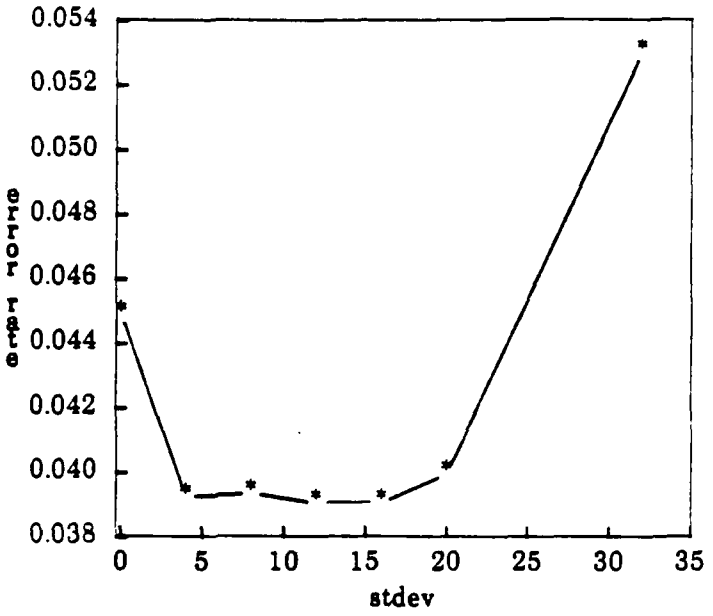
The optimal thresholds happened to be 220 for the Sobel, 750 for the Kirsch (coincidently), and 125 for the thinned Kirsch. The result of using these operators and thresholds on the standard deviation 12 artificial image (figure 18b) is shown in figure 36.



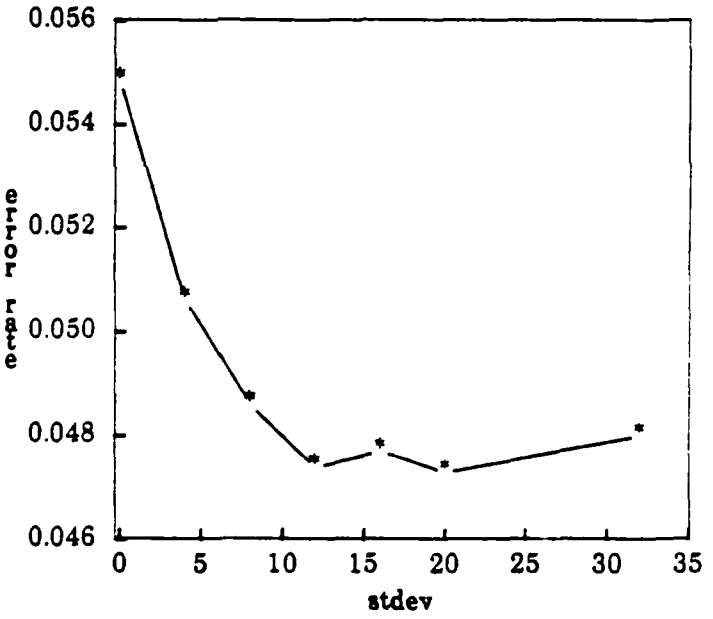
(a) Image with $\sigma = 12$ noise.
(b) Output of the Histogram Equalized Sobel Optimally Thresholded
(c) Output of the Histogram Equalized 5x5 Kirsch Optimally Thresholded
(d) Output of the Histogram Equalized Thinned Kirsch Optimally Thresholded
Figure 36: Application of Histogramed Equalized Operators to Artificial Image

In figure 37 are charts that describe the result of applying these operators to the histogram equalized artificial image (figure 18b).

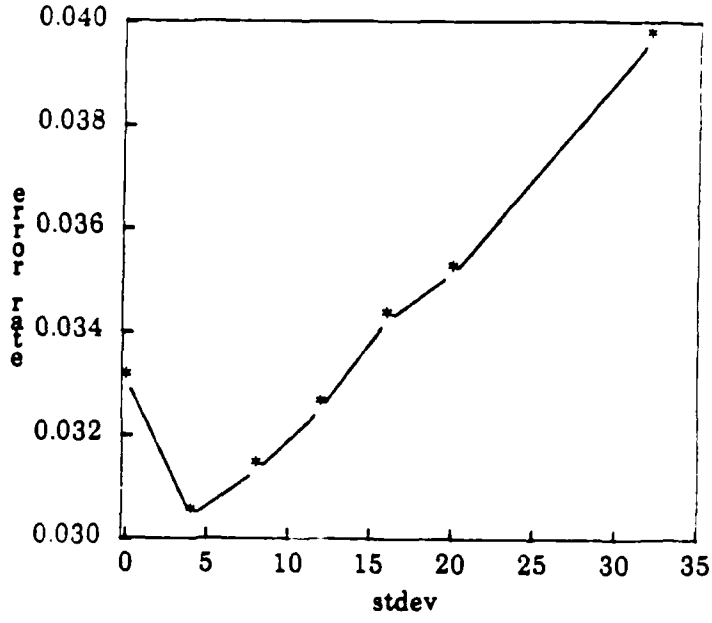
(a)


0.054
0.052
0.050
0.048
0.046
0.044
0.042
0.040
0.038
error rate
0
5
10
15
20
25
30
35
stdev

(b)

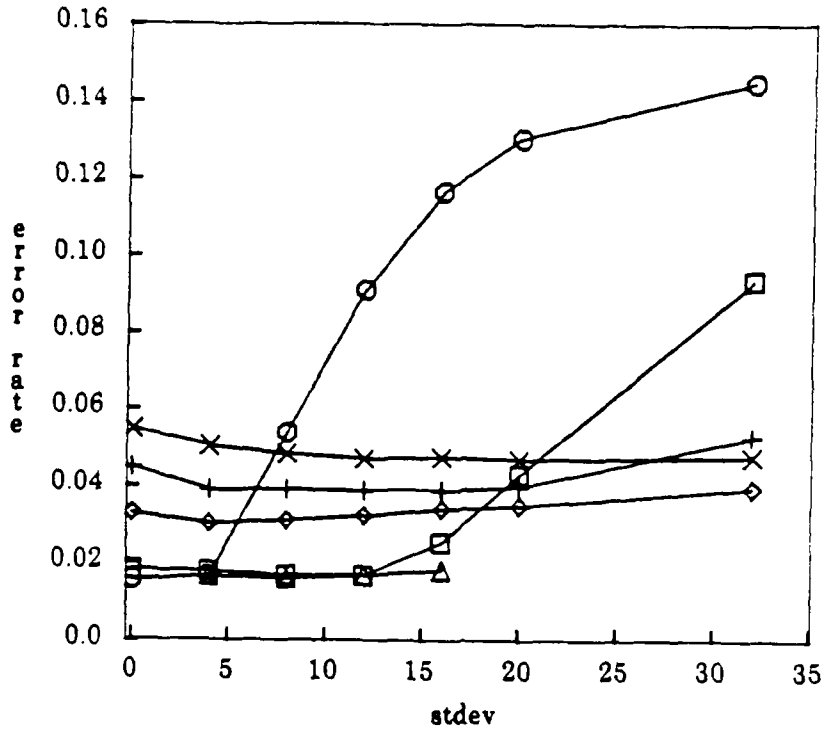

0.056
0.054
0.052
0.050
0.048
0.046
error rate
0
5
10
15
20
25
30
35
stdev

(c)



(a) Total Error rates for the thresholded Sobel on the Histogram Equalized Image
(b) Total Error rates for the thresholded Kirsch on the Histogram Equalized Image
(c) Total Error rates for the thinned Kirsch on the Histogram Equalized Image
Figure 37 Charts of Error Rates for Established Operators

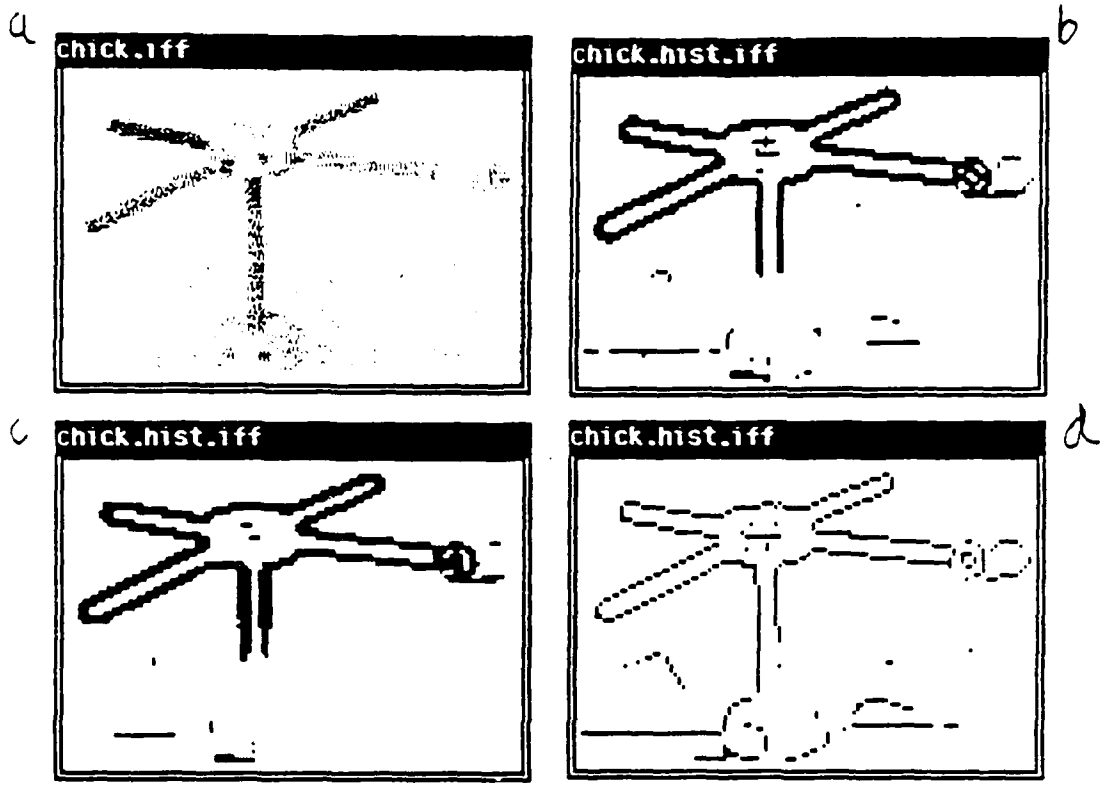
In figure 38 I compare the 3 operators on histogram equalized images to my operators on the original images (my operators do not expect histogram equalization and doing so may confuse them).



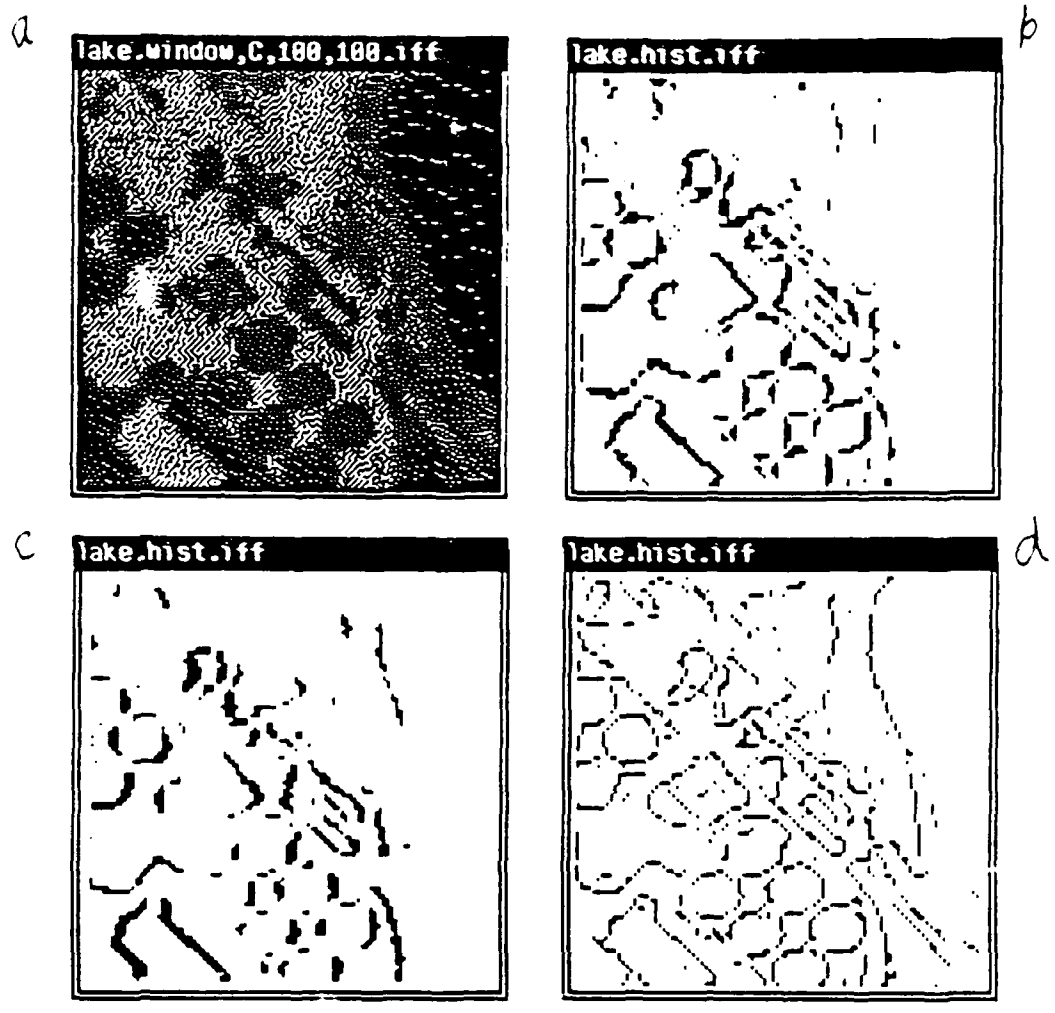
circles: Tuned $\sigma = 12$ operator
squares: Tuned $\sigma = 4$ operator
triangles: Tuned to noise σ operator
crosses: Histogram Equalized Sobel's error rate
X's: Histogram Equalized 5x5 Kirsch's error rate
diamonds: Histogram Equalized Thinned Kirsch's error rate

Figure 38: Comparison with Established Operators Applied to Histogram Equalized Image

Since the artificial images already had the full range of graylevels histogram equalization did not help the established operators much. However the advantage of histogram equalization is shown clearly when the operators are applied to real images in figures 39, and 40.



(a) Tinkertoy Image
(b) Output of the Histogram Equalized Sobel Optimally Thresholded
(c) Output of the Histogram Equalized 5x5 Kirsch Optimally Thresholded
(d) Output of the Histogram Equalized Thinned Kirsch Optimally Thresholded
Figure 39: Application of Established Operators to Tinkertoy Image



(a) Aerial Image
(b) Output of the Histogram Equalized Sobel Optimally Thresholded
(c) Output of the Histogram Equalized 5x5 Kirsch Optimally Thresholded
(d) Output of the Histogram Equalized Thinned Kirsch Optimally Thresholded
Figure 40: Application of Established Operators to Aerial Image

9. Previous Work

Edge detection is the established vision task that bears most closely on the boundary detection problem I describe here. Edge detection has been one of the earliest and most important tasks attempted by computer vision systems. Usually edge detection is described as a problem in image reconstruction.

Edge detection is often characterized as discovering the contrast in a region of the ideal image when there is an boundary between two constant intensity regions of the ideal image. Since I am not motivated by image reconstruction this task is not of particular interest for me. However often edge detection algorithms are used for boundary point detection. The idea is to accept as boundaries the pixels whose windows the detector considers to have high contrast.

The first work on edge detection was by Roberts who developed the Roberts edge operator to detect boundaries and corners of blocks. It was a simple convolution operator probably inspired by convolution based p-ttern matching. Since Roberts edge detection has been worked on by a large number of vision workers. Some of the operators were worked out in a somewhat ad hoc manner

as the Roberts was. The "best" and most common example of such operators is the Sobel edge operator

Many have worked on "optimal operators" where some model of edges is presented and the "best" function that fits a specified functional form. The definition of "best" and the functional form varies. Almost everyone who takes this approach limits the functional forms of edge operators to convolutions.

Hueckel [Hueckel71] considers convolutions over a disc on the image. He models edges as step edges with linear boundaries occurring at random places in the disk. His functions are limited to look at only certain specific Bessel coordinates (of an integral Fourier transform) that he has determined are useful for edge detection. He takes into account somewhat the possibility of two edges in the region. In a later paper [Hueckel73] Hueckel considers edges that are two parallel step edges a few pixels apart. He analyzes such edges the same way he analyzed the previous kind of edges.

At MIT starting with Marr [Marr82] there has been concentration on zero crossing based edge detection. The edge detectors they use are to locate edges at zero crossings of a Laplacian of a Gaussian. [Torre86] [Lunscher86b] [Lunscher86a] describe how such an edge detector is an approximation to a spline based operator that has maximal output at edges (compared to elsewhere). Such an operator also has been shown always to create connected boundaries.

Canny [Canny83] has examined the issue of convolution based edge detection more closely. In particular he studied the goals of edge detection. He considered an edge detector to be good if it reported strongly when there was an edge there and *did not report* when there was no edge. He also wanted a detector that only reported once for an edge. He found that these constraints conflict when one is limited to convolution based edge detectors (such behavior arises naturally for the boundary point detectors in this paper). His primary work on this topic was with a 1 dimensional step edge model. He derived a convolution operator that was similar to a 0 crossing operator. He also discusses how to extend the operators defined for one dimensional images to two dimensional images, and when oriented operators are desirable. His operators, applied to real images, usually appear to do a good job of finding the boundaries. In this paper I derive boundary point detectors for step edges but do not constrain the functional form of the edge detector. Thus the edge detectors based on this should have performance at least as good as Canny's detector.

Nalwa [Nalwa84] used a more sophisticated model where he assumed that regions in the intensity image fit (at least locally) surfaces that are planar, cubic or tanh type. He tested whether a surface fit a window on the image and if not he tried to fit various boundaries between surfaces. He ordered the tests to be of increasing computational complexity. His operators, applied to real images, usually appear to do a good job of finding the boundaries. The work in this paper handles models of this form and derives optimal operators.

Another approach to edge detection is to simulate parts of the human early visual system. Zero crossing operators were originally motivated by this argument since it was found that there were cells in the human early visual system that compute zero crossings at various frequencies [Marr82]. Other work that seriously studies the human early visual system was by Fleet [Fleet84] on the spatio-temporal properties of center-surrounds operators in the early human visual system. My work is not concerned with the structure of the early human visual system since its goals are to perform a task best as possible rather than as human-like as possible. However I can draw inspiration from the human visual system since it has been highly optimized to its goals by

evolution and hence an optimal detection system may be similar to that of the human eye (or animal eye for that matter).

Haralick has taken a similar approach to mine for the problem of edge detection [Haralick86a]. The differences between his approach and mine are that he models the image as a surface rather than as a function of a scene, and his operators generate decisions about edges rather than probabilities of edges [Haralick84]. However he has told me that his theory can be used to generate likelihoods that can be used with the techniques presented here [Haralick86b]. The relationship between his facet model and my template based models is currently under investigation.

10. Conclusion

I have demonstrated an operator that fulfills the desiderata in section 1. It has flexible output that can be used by many operators because it returns probabilities. It works on gray scale input. Because the operator is based on windows it does work proportional to the size of the image to calculate boundary probabilities. By constructing templates to represent a boundary shifted less than a pixel one can have subpixel precision with work proportional to that precision. A parameter of the algorithms I describe is the expected distribution of illuminances. Another is the standard deviation of and correlation in the noise.

Results were reported from using a 5 by 5 operator developed from this theory in section 8. I have applied this detector to artificial and real images. In section 8.3 I have compared my detectors to the established detectors, Sobel, Kirsch, and thinned Kirsch. In the next few weeks results from using a 7 by 7 and 9 by 9 operator will be available. Also comparisons will be done between these detectors and more advanced edge detectors such as Canny's [Canny83], and Haralick's [Haralick84].

In a companion report I describe an evidence combination theory that is applied to operators that return likelihoods that allows me to combine robustly the output of several different operators on the same data [Sher87]. Soon there will be results from using the likelihoods as input to a Markov random field based system [Chou87].

References

[Andrews77]
H. C. Andrews and B. R. Hunt, *Digital Image Restoration*, 8-26 , Prentice-Hall, INC.., Englewood Cliffs, New Jersey 07632, 1977

[Berger80] J. O. Berger, *Statistical Decision Theory*, 110-112 , Springer-Verlag., New York Heidelberg Berlin, 1980

[Binford81]
T. O. Binford, Inferring Surfaces from Images, *Artificial Intelligence 17*,1-3 (August 1981), 205-244, North-Holland Publishing Company.

[Boult6?] T. E. Boult and J. R. Kender, *On Visual Surface Reconstruction Using Sparse Depth Data*, Department of Computer Science, Columbia University., 1986?

[Brown82] C. M. Brown, Bias and Noise in the Hough Transform 1: theory, 105, Department of Computer Science, University of Rochester, June 1982.

[Canny83] J. F. Canny, Finding Edges and Lines in Images, 720, MIT Artificial Intelligence Laboratory, June 1983.

[Chou87] P. Chou and D. Sher, (Markov Random Fields for Information Fusion Based Segmentation) ?, To be Published in 87.

[Fleet84] D. J. Fleet, The Early Processing of Spatio - Temporal Visual Information, 84-7, University of Toronto, Research in Biological and COmputational Vision, September 1984.

[Geman84]
S. Geman and D. Geman, Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, *PAMI 6*,6 (Novermber 1984), 721-741, IEEE

[Haralick84]
R. M. Haralick, Digital Step Edges from Zero Crossing of Second Directional Derivatives, *PAMI 6*,1 (January 1984), 58-68, IEEE.

[Haralick86a]
R. M. Haralick, The Facet Approach to Gradient Edge Detection, *Tutorial 1 Facet Model Image Processing (CVPR)*, May 1986.

[Haralick86b]
R. Haralick, Personal Communication, June 1986.

[Hueckel71]
M. H. Hueckel, An Operator Which Locates Edges in Digitized Pictures, *Journal of the Association for Computing Machinery 18*,1 (January 1971), 113-125, ACM.

[Hueckel73]

M. H. Hueckel, A local Visual Operator Which Recognizes Edges and Lines, *J. ACM 20*,4 (October 1973), 634-647, ACM.

[Lunscher86a]

W. H. H. J. Lunscher and M. P. Beddoes, Optimal Edge Detector Design II: Coefficient Quantization, *Pattern Analysis and Machine Intelligence 8*,2 (March 1986), 178-187, IEEE.

[Lunscher86b]

W. H. H. J. Lunscher and M. P. Beddoes, Optimal Edge Detector Design I: Parameter Selection and Noise Effects, *Pattern Analysis and Machine Intelligence 8*,2 (March 1986), 164-177, IEEE.

[Marr82] D. Marr, *Vision*, W. H. Freeman and Company., New York, 1982

[Marroquin85]

J. L. Marroquin, Probabilistic Solution of Inverse Problems, Tech. Rep. 860, MIT Artificial Intelligence Laboratory, September 1985.

[Nalwa84] V. S. Nalwa, On Detecting Edges, *Proceedings: Image Understanding Workshop*, October 1984, 157-164.

[Sher85] D. B. Sher, Evidence Combination for Vision using Likelihood Generators, *Proceedings: Image Understanding Workshop (DARPA)*, Miami, Florida, December 1985, 255-270. Sponsored by: Information Processing Techniques Office Defence Advanced Research Projects Agency.

[Sher86] D. Sher, Optimal Likelihood Detectors for Boundary Detection Under Gaussian Additive Noise, *IEEE Conference on Computer Vision and Pattern Recognition*, Miami, Florida, June 1986.

[Sher87] D. B. Sher, Evidence Combination Based on Likelihood Generators, TR192, University of Rochester Computer Science Department, Milan, Italy, January 1987. Submitted in shorter form to IJCAI.

[Torre86] V. Torre and T. A. Poggio, On Edge Detection, *Pattern Analysis and MAchine Intelligence 8*,2 (March 1986), 147-163, IEEE.

Order and Structure in Stereo Correspondence

Paul R. Cooper
Department of Computer Science
The University of Rochester
Rochester, New York 14627

TR 216
June 1987

Abstract

Experiments with correspondence algorithms using dynamic programming have shown that robust and high quality correspondences can be computed, even between images acquired from radically different viewpoints, on different days, under different lighting conditions, and for scenes of discontinuous depth change. This paper examines the underlying reasons that these algorithms obtain such results.

The spatial structure of a scene and image can be represented by ordering image features by spatial position. Structure and its representation by order describe non-local characteristics of a scene and image which remain invariant to changes in viewpoint. Thus, order from structure provides an excellent mechanism for detecting correspondence between different images of a scene. Once such an ordering is established, dynamic programming provides one method for resolving remaining ambiguities.

The notion of correspondence from structure is developed by comparison with other underlying constraints discovered in prior work on correspondence. Some observations about the theoretical and experimental performance of correspondence algorithms based on order from structure are made.

1. Introduction

A problem of central interest to many vision researchers is detecting the correspondence between two or more different images of the same scene. The correspondence problem arises, for example, in stereo and motion processing, as well as during the verification phase in recognition by hypothesis-and-verification. Recent research has focussed on matching features extracted from the image which are invariant with respect to changes in factors such as viewpoint, lighting, and signal noise. Typically, edges are extracted and matched.

One principled way to frame the edge correspondence problem is as a combinatorial optimization. In this view, computing the correspondence between two images involves nxm pairwise comparisons of n edges from one image and m edges from the other. The problem is to select the subset of pairings that yields the best overall match. Unfortunately, there are $nm!$ many such subsets to choose from. Clearly, if we view the problem in this way, the challenge is to develop constraints which will allow us to efficiently select the best overall match.

Scene and image structure provide a powerful constraint for computing edge correspondence. Spatial structure can be represented as an ordering of features by relative spatial position. Correspondence of features from two images can be established by comparing similar orderings from each image. Furthermore, once such an ordering has been established, dynamic programming can be applied to resolve remaining ambiguity in the selection of the best overall match.

As experiments show, computing correspondence from spatial structure is very effective, even for problems involving changes in viewpoint, lighting conditions, and domain. This success derives from two main strengths not typical of other correspondence algorithms. First, structure is highly non-local in nature. Second, in most correspondence problems, relational structure is invariant.

As the discussion of correspondence by structure is developed, the importance of these two traits will become clear. First though, the paper spends some time tracing the discovery of the central difficulties in the correspondence problem, and the constraints that were developed to address them. After providing this perspective, the paper describes structure and its representation by spatial ordering. The advantages of structure for correspondence are described, and its use in a dynamic

programming framework is outlined. Finally, some experiments are presented, demonstrating the effectiveness of the approach.

2. Local Analysis and Constraints

Originally, the correspondence problem was thought to be simply an issue of image intensity correlation. Addressing local position variation (in stereo and motion correspondence) required modifying this approach to matched filtering for point or patch characteristics [Hannah 1974, Panton 1981]. Such intensity matching approaches were soon found susceptible to failure, due to variation in the intensity signal introduced by factors such as signal noise and changes in viewpoint. This motivated the development of matching features extracted from the signal, the hope being that such features represented real invariants in the scene. Typically, edges are detected and matched.

The local patch matching heritage evolved into a local edge matching process. In principle, each local edge from one image could match to any of the many edges in the other image. The local characteristics of an edge (its direction and the polarity of its intensity change) are available for discrimination. But the most important constraints derive from an edge's *position*. Some positional constraints are derived from imaging geometry, as opposed to scene geometry. Thus, epipolarity constrains a local edges match to lie along a 2D line in the other image [Barnard and Fischler 1982]. Other positional constraints are derived from assumptions about the world. One can assume smoothness, or guess maximum depth ranges allowable, and constrain the positional range in which a match might occur in that fashion.

Unfortunately, even after all these constraints have been applied, ambiguity fo matching often remains. This is due to a basic deficiency of local analysis: it suffers from the aperture problem, or "keyhole" effect. As if peering through a keyhole, local match search is ignorant about the non-local structure of the scene which could resolve apparent local ambiguity. Furthermore, local match analysis by position often lacks a coherent procedural framework. Algorithms typically operate by "looking around" in the same place in the other image for a match [Grimson 1981], and assuming no ambiguity will occur.

3. Non-Local Analysis and Constraints

Some significant improvements have been developed however, most of which address "keyhole" effect problems. The most successful strategy, now an accepted tool in perceptual work of all kinds, is coarse-to-fine multi-resolution analysis [Marr 1982], or as recently generalized, scale-space analysis[Witkin 1986, Witkin et al 1987]. The strength of this notion results from the fact that it addresses the two main problems of correspondence simultaneously - coarse scale analysis is not only less local (and therefore less ambiguous), it is less sensitive to signal noise as well. One important characteristic of scale-space analysis is that it arises from controlling the *imaging process*, not from building knowledge of the scene.

Another constraint, sometimes called the constraint of "figural continuity" also counteracts the "keyhole" effect [Mayhew and Frisby 1981, Ohta and Kanade 1985, Cooper et al 1987, Grimson 1985]. The strength of this constraint is due to the non-local and less ambiguous nature of a connected contour, which reflects invariant and real structure in the world.

3.1 Dynamic Programming

Even more progress in addressing the correspondence problem was achieved with the introduction of dynamic programming as a framework for match search [Baker 198?] The algorithm [Bellman 1957] is designed to efficiently compute a global optimum (in this case the best overall edge match). With the addition of one constraint, it is well suited for the edge correspondence problem as framed in the introduction. The essence of a dynamic programming algorithm for matching is the recurrence relation:

$$GS(i,j) = LS(i,j) + \max_{k,l \in R} GS(k,l) \qquad \text{[equation 1]}$$

where

- GS(i,j) is the global score if 'i matches j' is part of the best overall match
- LS(i,j) is the local score computed by comparing features i and j
- $\max_{k,l \in R} GS(k,l)$ is the maximum of global scores prior to i,j in the region R.
  In the standard formulation, R = {(i-1, j), (i-1, j-1), (i, j-1)}

As originally designed, dynamic programming optimized decisions ocurring over time (thus the "dynamic" modifier). Assumed was that the events being compared in the LS predicate were *ordered*.

Baker[1982] observed that edges along corresponding epipolar lines maintain such an order, left-to-right, allowing computation of line-to-line correspondence by dynamic programming. Ohta and Kanade[1985] extended the framework so it could handle connected contours (incorporating the "figural continuity" constraint). To accomplish this, the dynamic programming recurrence relation had to select the best path from a 3D cube instead of a 2D table. Also, a spatial sort of contours was performed prior to the application of the dynamic programming equation, in order to establish the required ordering. Both Baker's and Ohta and Kanade's algorithms were reported very successful.

4. Structure and Order

The reason correspondence by dynamic programming works well is that the edge ordering represents the underlying invariant non-local structure of the real scene. As will be seen, this is true to such an extreme extent that once a good ordering has been obtained from each image, the dynamic programming selection of best match is almost irrelevant.

4.1 Structure

"Structure" is defined here to be a description of the relative spatial position of entities. Such descriptions are stratified and hierarchical. Thus, scene structure is the relative position of the objects in a scene. Object structure refers to the spatial composition of an object with respect to its sub-components. And sub-components can have structure as well.

Image structure is the image projection of scene and object structure. As Witkin and Tenenbaum [1983] note, the most easily detectable image structure is non-accidental regularities in the image which arise from the projection of primitive component structure. More generally, if contours of connected edges arise from the projection of invariant boundary structure in the scene, the relative spatial position of contours in the image is the image structure corresponding to the scene and object structure.

4.2 Structure is Non-Local and Unambiguous

Obviously enough, structure is non-local. In the scene, the relative position of objects and components to each other is highly non-local. Furthermore, with respect to any one reference frame, these positional relationships are unique. In the image, the position of one contour with repect to all others in the image is unique and non-local also.

4.3 The Invariance of Qualitative Structure

In many circumstances, the same structural relationships hold in two different images of a scene. To begin with, if we consider only static scenes, we know that the scene structure is unchanging. If the scene structure is invariant, over a wide range of viewpoints the *qualitative* relationships constituting structure in an image remain the same. For example, in a scene where a dog is "to the right of" a cat, over a wide range of viewpoints the dog and cat have the same qualitative spatial relationship to each other. (In fact, the dog and cat maintain the same relationship over nearly half of all possible viewpoints. Only by viewing the cat and dog from the completely inverse direction is the relationship inverted). When the components of an *object* maintain this invariance over a range of viewpoints, this is sometimes represented explicitly and referred to as the principal view [Feldman 1985] or generic or characteristic view of the object [Chakravarty 1982].

Clearly, the exact *quantitative* relationship between the dog and cat does vary in images acquired from different viewpoints. ("How much" the dog is to the right of the cat in the image changes).

Of course few invariants remain unchanging under all circumstances. The utility of an invariant depends on the fraction of expected cases over which it holds, criteria hard to quantify and evaluate. For correspondence, there is a small viewpoint range over which invariants can be reasonably expected to hold (not including, for example, both the front and back views of the dog and cat). Within this range, there is only one major difficulty for the invariance of spatial image structure. This arises when two objects are at greatly different distances from the imaging device, and aligned in view. (If the nearer object is large, it occludes the more distant object under these cicumstances). This situation is highly degenerate with respect to viewpoint. Thus, a small change in viewpoint can result in a change in the relative

relationship between the objects in the two images. It is important to note that even under these circumstances, the invariance is only violated locally. The position of a contour with respect to almost everything else in the image still remains the same. (It is also interesting to note that this situation virtually never arises with single connected objects; structural invariance remains inviolate for principal views).

If the structure of the scene is varying (ie. if the objects are in motion with respect to each other) structural invariance in the images cannot be expected to hold, strictly speaking. On the other hand, even in this situation, most of the scene remains structurally invariant.

The invariance of image structure depends as little as possible upon assumptions about the world. Objects with boundaries continuous over scale larger than a pixel must exist somewhere in the scene. More general assumptions of smoothness are not required.

4.4 Spatial Order Represents Structure

One efficient representation of qualitative image structure is an ordering of image features generated by sorting the features by spatial position. For example, the relative positions of contours extracted from an image can be represented by sorting them by spatial position. The image coordinate system implicitly provides one convenient reference frame.

There is no obviously natural way to order connected contours in a 2D image. However, any ordering will suffice, provided it preserves invariance [Yuille and Poggio 1984]. Ohta and Kanade [1985] developed the first such sort for connected contours, basing it on the 1D invariance of left-to-right precedence along scan lines in the image. Figure 1 shows the order assigned to some example contours by this sorting algorithm. The key thing to note is that a contour's position in the sequence represents its spatial relationship with respect to all other contours in the image. For example, in Figure 1, contour d is below and to the right of contour a.

Clearly, representing structure with order captures some, but not all, of the structure information inherent in the positions of the contours. For starters, no quantitative information is represented; "how much" contour d is below and right of contour a is lost. This is advantageous - only the invariant qualitative structure is captured by the ordering representation. In addition, the detailed information about

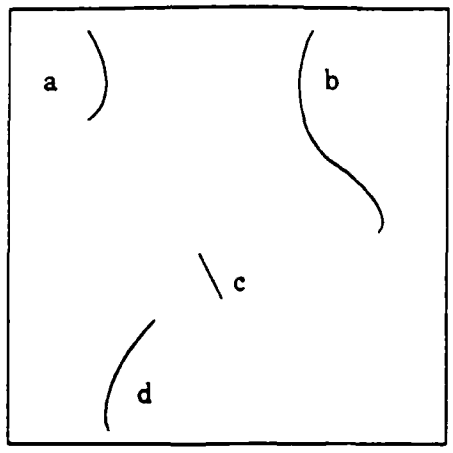


Figure 1: Sorting Contours by Spatial Position

Order from Ohta and Kanade's [1985] sort:
a c d b

the relative positions of contours is compressed into one relation - "before" and "after".

Representing structure by spatial order trades off detail for efficiency. A single number provides an unambiguous representation of the position of a feature, non-locally relative to all others in the image. The fundamental spatial organization of the image, which reflects the structure of the scene, is captured compactly.

5. Stereo Correspondence from Order Representing Structure

5.1 In Theory

In theory, computing correspondence from structure is quite straightforward. One simple algorithm is as follows. Connected edge contours are extracted from both left and right images. The contours from each image are sorted by spatial position, generating a labelling sequence for each image. A contour's position in the sequence assigns it a unique label for that image. If structural invariance holds, the matching contour in the other image is assigned the same label in the sequence for that image. In this way, the sequences from the spatial sorting process determine the correct match for every contour in the stereo pair.

Thus, under ideal circumstances, the spatial sorting process *alone* is sufficient to compute correspondence. Furthermore, this matching algorithm is not susceptible to ambiguity associated with local "keyhole" analysis. Only real conditions of

degeneracy in the world and imaging geometry can degrade the quality of the correspondence (as described earlier). In addition, determining correspondence in this way is computationally efficient. The complexity of this algorithm is a function only of the cost of sorting the contours.

If spatial sorting of the contours is sufficient by itself to compute correspondence, a subsequent computationally expensive dynamic programming process (such as that in Ohta and Kanade's [1985] correspondence algorithm) is superfluous. However, in less than ideal circumstances, the labelling sequences generated by the spatial sorting process will not be exactly the same for the left and right images. In this case, the ambiguity which does remain must be resolved, and the best overall match of one sequence to the other must be computed. If order invariance can be maintained, dynamic programming provides one way of computing this best sequence match.

If a dynamic programming process is to be used to match the two sequences, the simplest formulation is exactly that of equation 1. This represents a considerable simplification from Ohta and Kanade's [1985] dynamic programming match. Instead of a 3D dynamic programming process, only one simple and much more efficient 2D table is used. No line-to-line correspondences need be computed. Instead, the correspondence is computed from the non-local, invariant, structure-derived ordering of the connected contours directly.

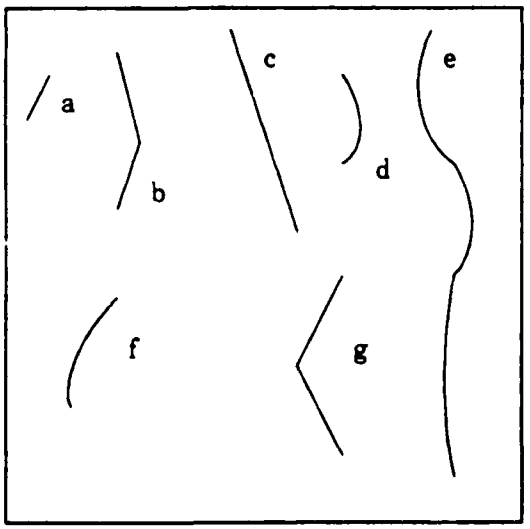
5.2 In Practice

In practice, difficulties arise which make the process more complex. The main difficulty is due to signal noise and lighting change. In the face of real signals with real signal noise, complete invariance of feature extraction can never be guaranteed. Clearly, if the features themselves change from one instantiation to another, representing their relationships invariantly is difficult. Therefore, the quality of correspondence generated depends upon how robust the sorting and selection process can be made. The sequence order in particular should be made as invariant as possible to the effects of noise.

Consider the problem of an "extra" edge. The presence of noise implies the possibility that an edge (or contour) might be detected in one image but not the other. For sequences generated from simple one dimensional spatial sorts along

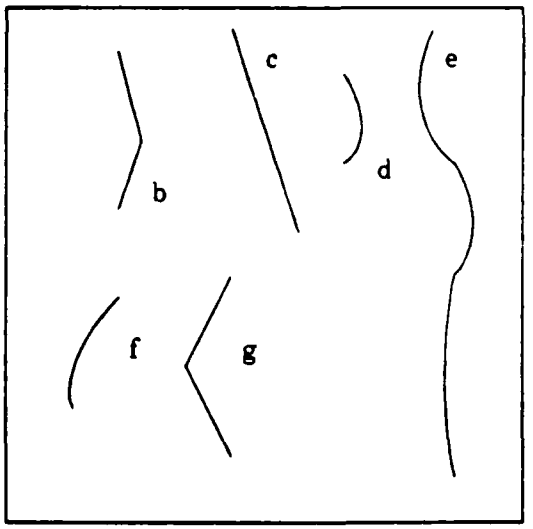
scan lines (as used, for example, in [Baker 1982]) the ordering relationship between the edges remains the same in each image, even though edges may be deleted or added. The consequences of an "extra" edge for the increasingly non-local sort used by Ohta and Kanade [1985] are much more severe. Figure 2 shows how the removal of a single "noise" edge from the set of contours radically alters the order generated from Ohta and Kanade's sort. Under these circumstances, it becomes more difficult to compute correspondence with dynamic programming based algorithms. This situation can arise because Ohta and Kanade's sort is indifferent to the order of contours with no left-to-right dependence along a scan line.

It is possible to develop spatial sorting algorithms which are more robust against effects of noise such as these. Ohta and Kanade's spatial sort is based on the invariance of left-to-right ordering of contours along epipolar lines in a stereo pair. But an invariant spatial relationship between contours is also present in the vertical direction. An alternative sort which takes advantage of this fact uses top-to-bottom order as a sort "key" as well. That is, both left-to-right precedence and top-to-bottom precedence are used to order the contours. On a pairwise basis, these criteria are mutually exclusive. But because of the transitivity of the left-of relation, conflict



Left Image

Order from Ohta and Kanade's [1985] sort: a f b g c d e
Order from alternate sort: a b c d f g e



Right Image
(contour a isn't present)

Order from Ohta and Kanade's [1985] sort: b f c g d e
Order from alternate sort: b c d f g e

Fig. 2: Spatial Sorting with Noise Contours

between the two precedences is possible. In this case, the left-to-right precedence is preferred. Example orderings generated by such a sort are given in Figure 2. Note that the order which is computed remains invariant in the presence of insignificant local change between the images.

The adoption of additional strategies to address the effects of noise can make order-based correspondence algorithms even more robust. For example, provision must be made in the dynamic programming process for allowing multiple matches of contours from the left image to a single contour in the right image, in case the connectivity of extracted contours varies between the images. As in line-to-line matching algorithms, provision must be made for skipping matches entirely. In addition, incorporating coarse-to-fine analysis is a significant and proven strategy for obtaining better correspondence results in the presence of noise.

6. Experiments

Because of their use of sequences generated by spatial ordering, the dynamic programming based systems of Baker [1982] and Ohta and Kanade [1985] can be seen as algorithms which match structure, with it's non-local invariant properties. In both cases, however, the emphasis was upon line-to-line matching and the dynamic programming search process itself.

Cooper et al [1987] built a system which depends more directly upon extracting, representing and matching non-local invariant spatial structure over the entire image. This system uses both the modified sort described above, and the simplified 2D dynamic programming table which matches connected contours directly, independent of line-to-line matches. In addition, it incorporates coarse-to-fine multi-resolution analysis. In all, this system uses three major principles to specifically address the problem of local ambiguity in correspondence: invariant non-local structure-derived order, figural continuity, and coarse-to-fine analysis.

In this section, the results of three radically different experiments with this system are presented. The experiments were chosen to demonstrate the strength of the principle of matching from invariant non-local structure.

6.1 Satellite Images

Figures 3 though 6 show results of a previous experiment with this system, taken

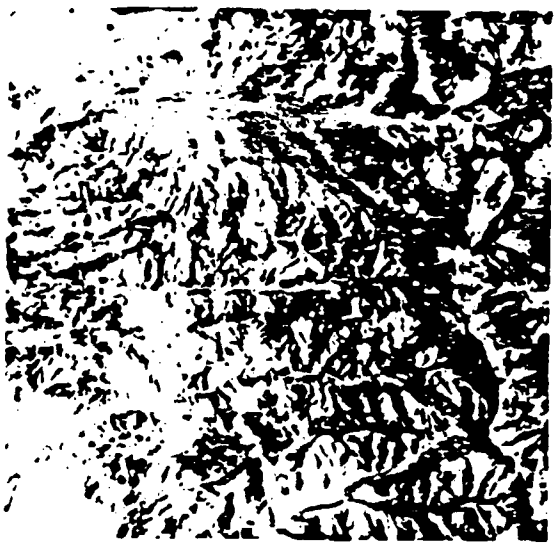


Figure 3: Death Valley Scene -
Original Satellite Image

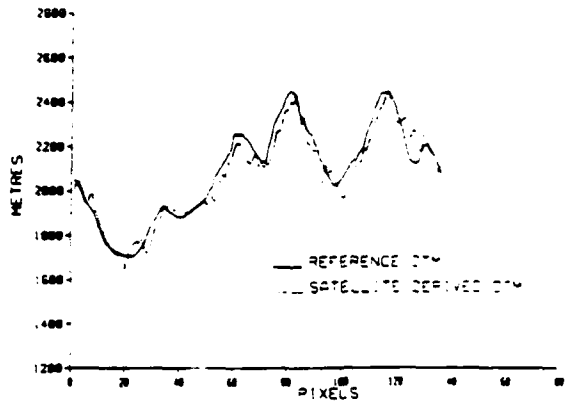


Figure 4: Death Valley Scene -
Comparative Profiles Through Depth Maps

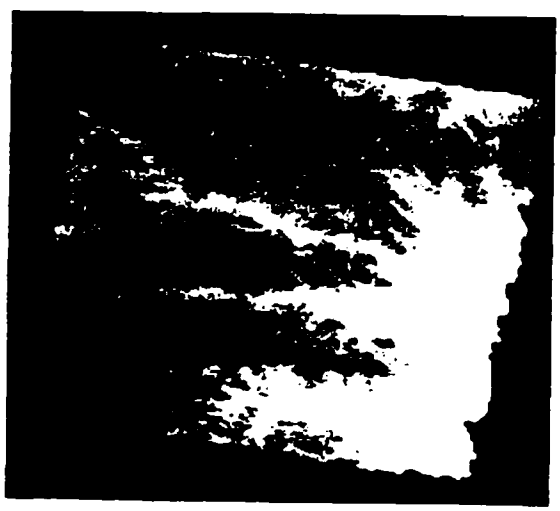


Figure 5: Death Valley Scene -
Stereo-Derived Digital Terrain Map

Fig 6: Death Valley Scene -
Reference Digital Terrain Map

from Cooper et al [1987]. In this experiment, real depths were generated from stereo satellite images and compared against reference digital terrain maps. Figure 3 shows one of the original images, a Landsat image of the Death Valley area in California. Figure 4 shows profiles extracted from both the reference depth map and the stereo-derived depth map (along a 'vertical' line running approximately through the middle of the scene). Figures 5 and 6 show the stereo-derived and reference depth maps in their entirety, with depth represented by intensity. The stereo pair in this case was acquired by the same satellite from radically different viewpoints, on different days, and under different lighting conditions. Over the entire scene, the RMS error between the reference map and the correspondence-derived depths was less than 60 meters. The quality of this result is particularly surprising given the resolution of the images, and the standard error of the reference map (60 meters). Further details and other results of this type are given in Cooper et al [1987].

6.2 Tinker Toy Domain

Figures 7 through 12 show the result of applying correspondence from structure to the Tinker Toy domain [Cooper and Hollbach 1987]. Figures 7 and 8 show the left and right images from the stereo pair, acquired by stereo CCD cameras in the vision lab of the University of Rochester. (The images as presented can be perceived in 3D). The object, constructed of Tinker Toys, is a model of the dinosaur Tyrannosaurus Rex. This particular test case is characterized by sharply discontinuous depths varying over a wide range, as well as the presence of surfaces (and contours) in one images which are occluded in the other image. (See, for example, the side of the object's head in the left image). The range of disparities in this experiment is 47 pixels, consituting about 1/3 of the width of the image of the object. This provides a particularly severe test of the correspondence process, as witnessed by the different tip from vertical of the object in each image.

Figures 9 and 10 are the connected contours extracted from the left and right images, respectively. A total of 46 distinct contours were detected in the left image, and 38 contours were detected in the right image. Note that the sets of contours are not identical, although nearly all of the important contours have been detected and connected correctly in both images. (Horizontal contours, from which it is difficult to extract depths when matched, are not shown in these figures).

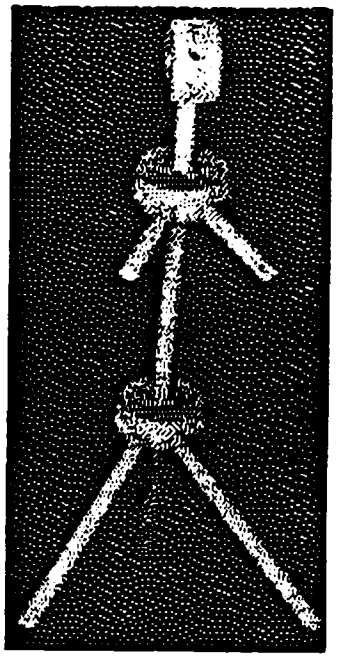


Figure 7: T. Rex Scene - Left Image of Tinker Toy Object

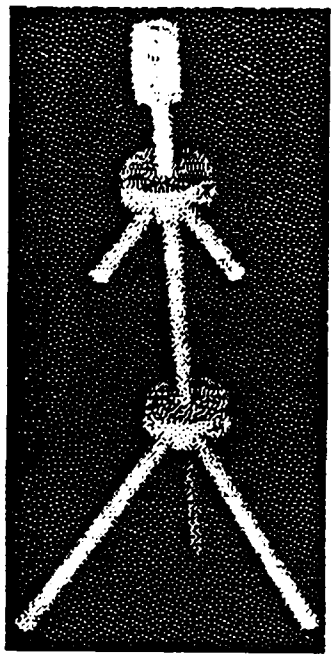


Figure 8: T. Rex Scene - Right Image

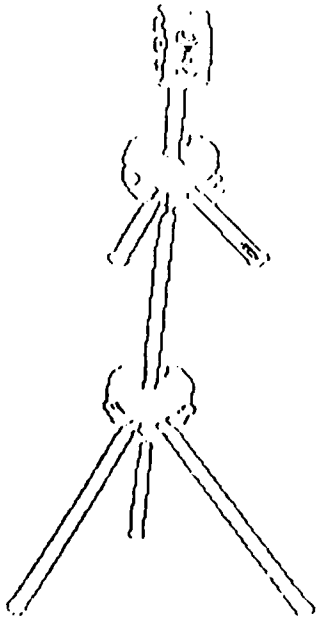


Figure 9: T. Rex Scene - Connected Contours Extracted from Left Image

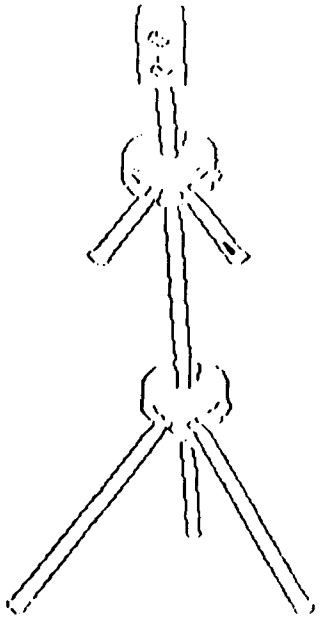


Figure 10: T. Rex Scene - Connected Contours Extracted from Right Image

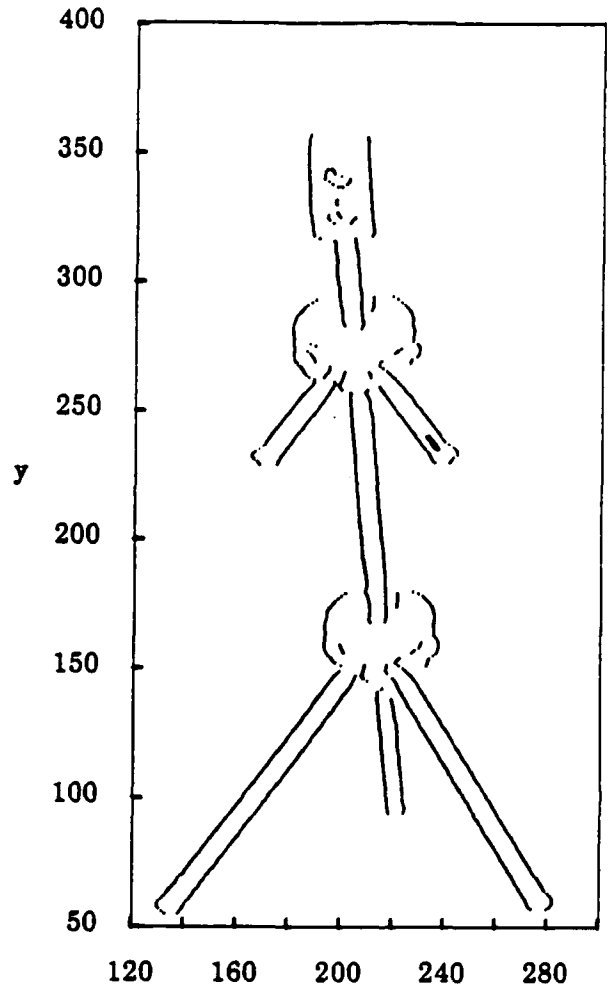


Figure 11: T. Rex Scene -
Plot of Matched Contours

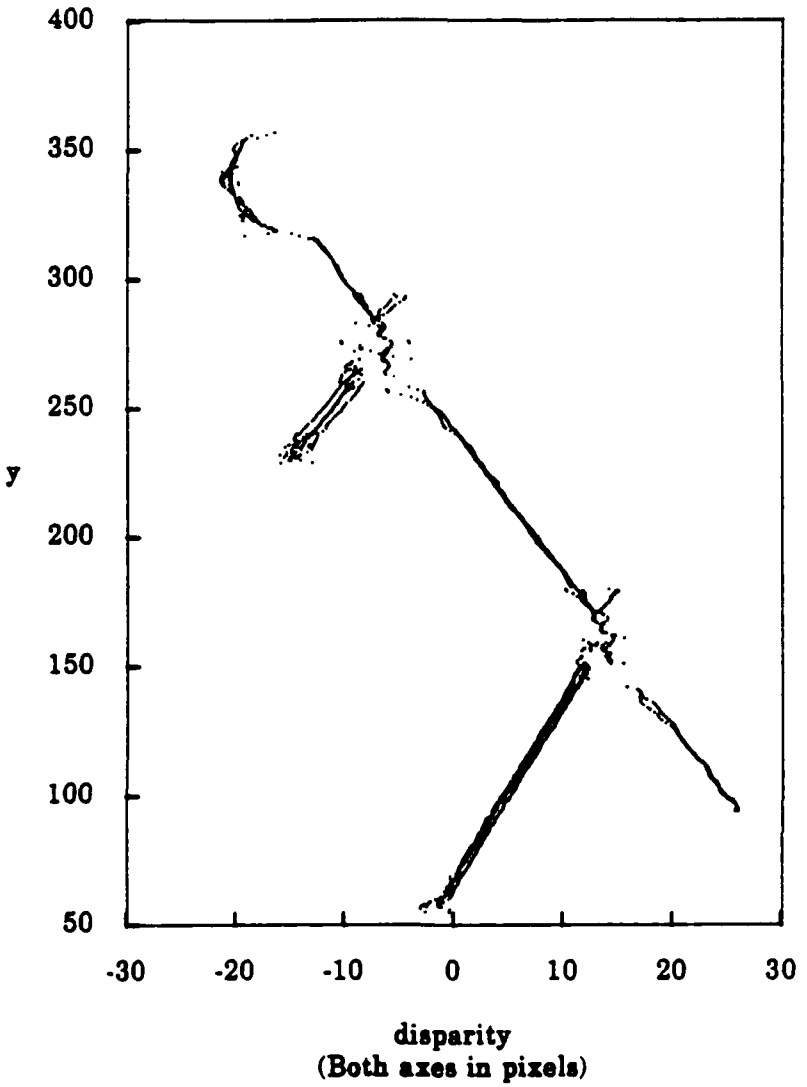


Figure 12: T. Rex Scene -
Side View of Matched Contours
Showing Disparities Computed
by Stereo

The correspondence process matches the connected contours. An x-y plot of the contours which were matched in the this case is given in Figure 11. (The matched contours are given in right image coordinates). As can be seen if Figure 11 is compared with Figure 10, each of the 38 contours is assigned a match along most of its length. All 38 matches were correctly determined in this case.

Finally, the accuracy of the disparity results which were computed from the correspondence can be seen in Figure 12. This figure is effectively a "side view" of the matched contours. The central axes of the body show continous depth variation, and the curvature of the head (a Tinker Toy "disk" at right angles to the cameras) is evident. The correspondence was obviously not confused by the abrupt depth differences between, for example, the forward projecting legs and the rear pointing tail.

6.3 Structured Light Experiment

Figures 13 and 14 show the stereo pair for a possibly even more challenging correspondence experiment. In this test, structured light consisting of parallel white stripes was *projected onto* simple geometric objects in a scene. (Once again the images were acquired by stereo CCD cameras mounted on the robotic 'head' in the vision lab at the University of Rochester). The object in the foreground is a sphere on a cylindrical mount, and in the background is a planar surface tipped so that it's depth varies from left to right throughout the scene.

The most interesting aspect of this experiment is the periodic nature of the contour features. Any matching algorithm which depends only upon a local analysis of these images will face unresolvable ambiguity in attempting to match the stripes (or contours), because of their periodicity. Furthermore, the disparity range over which matching must occur is again rather severe - a total of 32 pixels in this case. In fact, as can be observed when the two images are viewed in overlay, the disparity range is large enough that the correctly corresponding light stripes are offset by one entire stripe in some places. (In effect, the phase of the match is offset by one cycle at some places in the scene). There is also once again abrupt depth discontinuity present in the scene, as the sphere's edge constitutes an occluding boundary relative to the background plane. All in all, the experiment represents a near worst case test

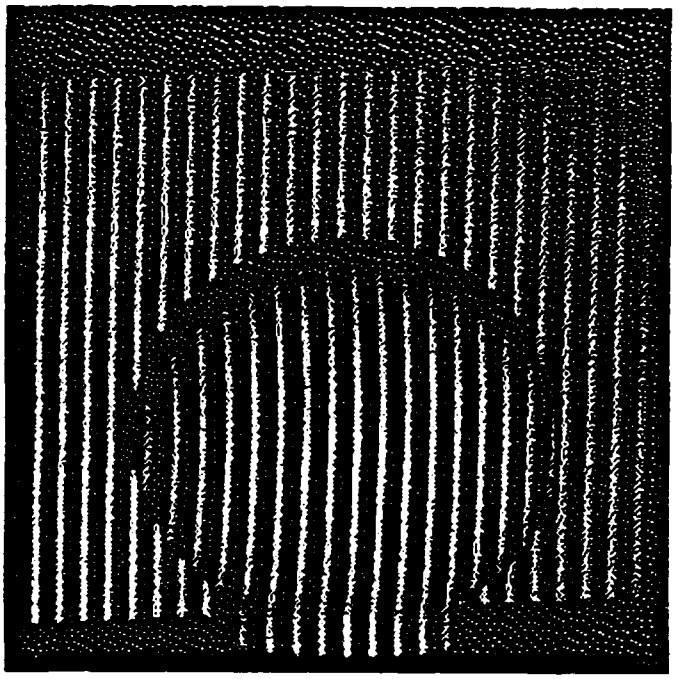


Figure 13: Structured Light Scene - Left Image

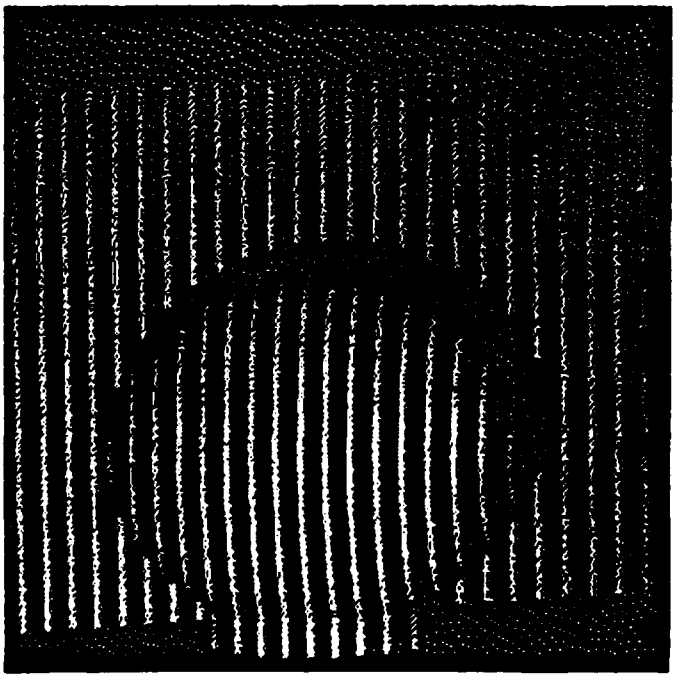


Figure 14: Structured Light Scene - Right Image

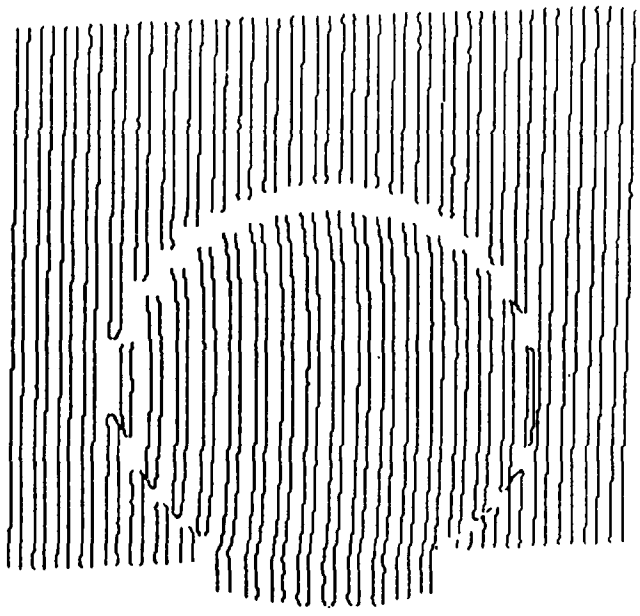


Figure 15: Structured Light Scene -
Connected Contours from Left Image

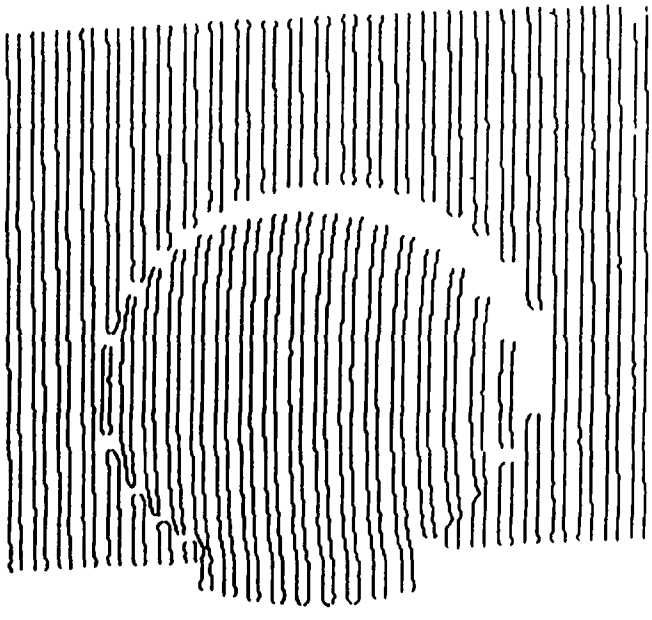


Figure 16: Structured Light Scene -
Connected Contours from Right Image

for most correspondence algorithms. As will be seen, matching non-local invariant structure-derived sequences is sufficient to address these difficulties.

Figures 15 and 16 show the extracted and linked contours for the left and right images. In all cases, each edge of a stripe was detected and linked over it's entire continuous length in the scene. A particularly impressive case is found at the right hand side of the sphere in the left image (Figure 13), where the stripes are detected separately within a few pixels of each other across the occluding depth discontinuity.

Figures 17 and 18 show a presentation of the disparities which were computed by the correspondence process. Because of the nature of the scene in this test case, the most convenient presentation of the results was obtained by generating an interpolated surface from the matched contours. The "S" statistics package [Becker and Chambers 1984] was used to generate the interpolated surface from the matched contours, as well as the perspective and contour plots of the surface shown in Figures 17 and 18. As can be observed, the correspondence computed by the matching algorithm was very nearly perfect. For example, the stripes on both sides of the occluding boundary on the right side of the sphere were correctly matched, yielding an impressive depth discontinuity in the interpolated surface.

One aspect of the experimental results is particularly interesting. After the spatial sorting process, a large fraction of the correct matches typically appear on or near the diagonal of the dynamic programming table. For example, consider the structured light experiment. Figures 15 and 16 show 100 and 97 distinct contours extracted from the left and right images, respectively. The dynamic programming table therefore contains about 10,000 or $O(n^2)$ elements in this case, each of which corresponds to a particular contour pairing. Furthermore, each of these potential pairings looks equally good, based upon a purely local evaluation. (Because of the repetitive nature of the test case, the local characteristics of the contours are largely the same. Most contours have the same shape and contrast, and even share the same epipolar line coordinates). Therefore, as far as an uninformed dynamic progamming process is concerned, a vast majority of the 10,000 potential pairings must be presumed to be candidates for the best overall match.

But following the spatial sorting process, all of the correct matches are found on or near the diagonal of the table. (During the actual experiment, the correspondence process assigned 96 matches, all of which were on or near the diagonal). In general,

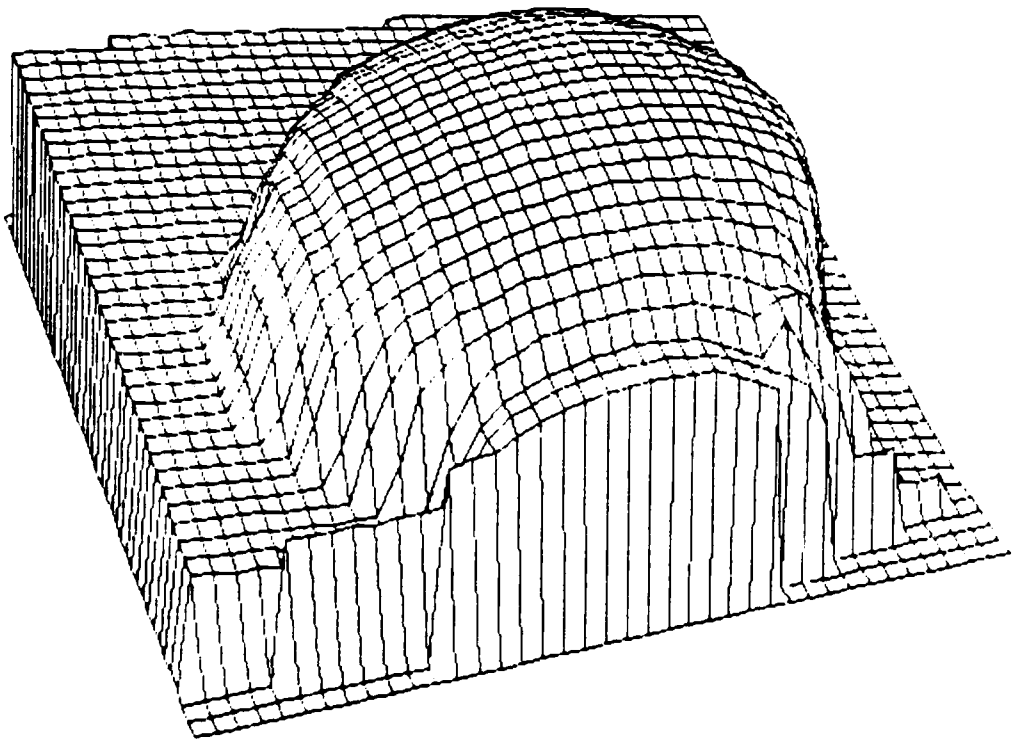


Figure 17: Structured Light Scene - Perspective View of Interpolated Disparity Surface

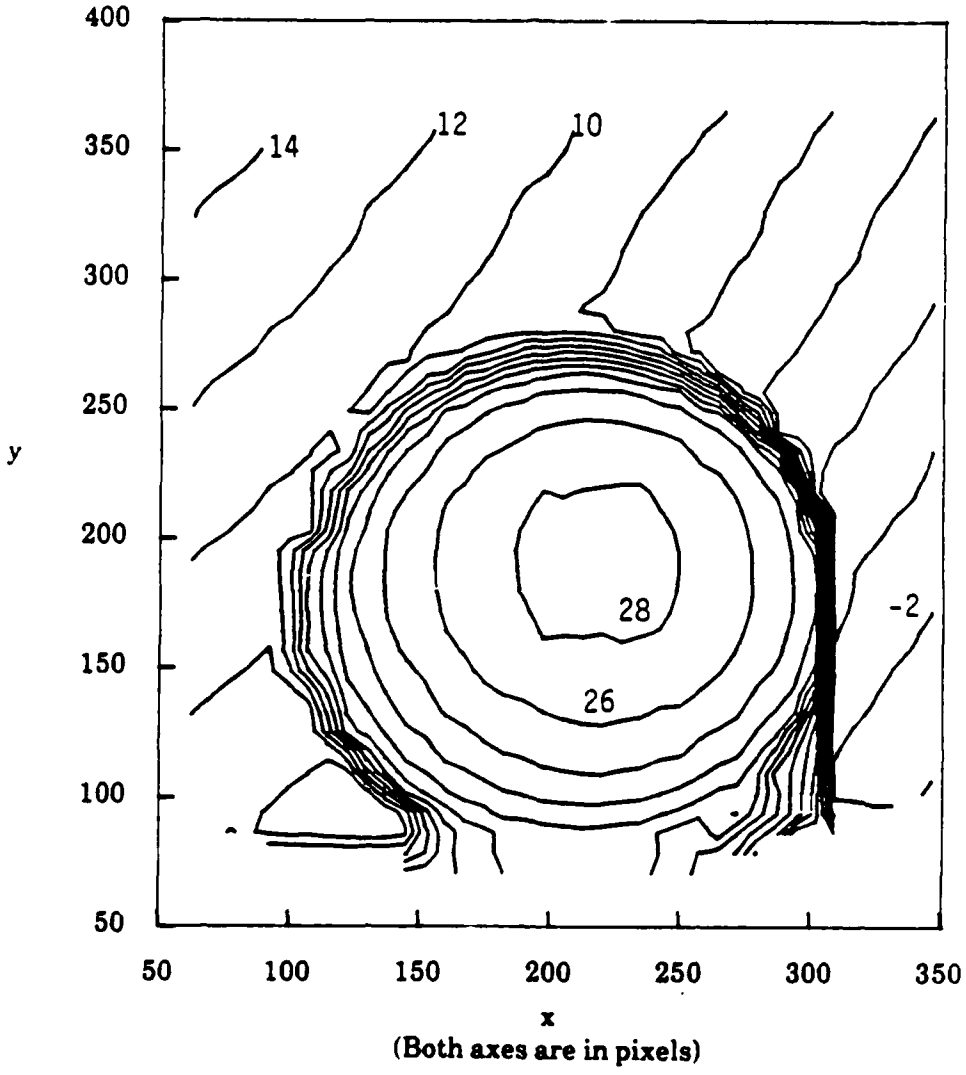


Figure 18: Structured Light Scene -
Contour Plot of Interpolated Disparity Surface

(Lines are contours of constant disparity)

the region on or near the diagonal constitutes only $O(n)$ of the potential pairings (only a few hundred in this case!), rather than $O(n^2)$. This indicates the extent to which the correspondence has been determined by the sorting process alone. The dynamic programming process is serving largely to resolve the minor ambiguities which remain following the sort.

As other experients have revealed, it is also true that once an ordering has been established, minor changes in the local score computation have little significant effect on the overall correspondence that is computed. On the other hand, minor changes to the ordering mechanism result in dramatic changes in the correspondence which is computed. This further emphasizes the importance of the structure-derived order in the correspondence computation.

6. Conclusions

This paper has addressed questions of *why* good correspondence algorithms work well. There are at least two major underlying problems in computing image correspondence - "keyhole effect" ambiguity due to local analysis, and variation due to noise and viewpoint change. To compensate for the latter problem, invariant features such as edges and contours are extracted from image intensity data and matched in a coarse-to-fine or scale-space framework.

The central contribution of this paper has been to demonstrate that extracting and representing the spatial structure of a scene provides a very powerful way to remove local match ambiguity and compute robust and high quality correspondences. Image structure can be effectively represented with an ordering of image features based on spatial position. For correspondence, representing structure by order in this way has two main strengths: it is non-local in nature, and it is invariant with respect to viewpoint change and signal noise. Representing structure with order also allows dynamic programming to be used as a framework to resolve remaining ambiguity in best match selection. In practice however, this dynamic programming stage is nearly superfluous.

Acknowledgements

The guidance and support provided by Jerry Feldman and Dan Friedmann during the development of these ideas is gratefully acknowledged.

This work was supported originally by an IRAP grant from the Canadian National Research Council to MacDonald, Dettwiler, and Associates of Vancouver, Canada. It was also supported in part by NSF Coordinated Experimental Research grant no. DCR-8320136, and in part by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700 and the Air Force Office of Scientific Research, Bolling AFB, DC 20332 under Contract No. F30602-85-C-0008. The latter contract supports the Northeast Artificial Intelligence Consortium (NAIC). I thank the Xerox Corporation University Grants Program for providing equipment used in the preparation of this paper.

References

Arnold, R.D. and T. Binford. "Geometric Constraints in Stereo Vision". *Proceedings, SPIE Vol. 238*, 1982.

Baker, Harlyn H. *Depth from Edge and Intensity Based Stereo.* Stanford University AIM 347, Stanford, California, 1982.

Barnard, S.T. and M.A. Fischler. "Computational Stereo". *ACM Computing Surveys*, 14(4), 553-572, 1982.

Ballard, D. and C. Brown. *Computer Vision*, Englewood Cliffs, N.J.: Prentice Hall, 1982.

Becker, R. A. and J. M. Chambers. *S: An Interactive Environment for Data Analysis and Graphics.* Belmont, CA: Wadsworth, Inc. 1984.

Bellman, Richard. *Dynamic Programming.* Princeton: Princeton University Press, 1957.

Chakravarty, I. and H. Freeman. "Characteristic Views as a basis for three dimensional object recognition", *SPIE Vol. 336* (Robot Vision), 37-45, 1982.

Cooper, Paul R., Daniel E. Friedmann and Scott A. Wood. 1985. "The Automatic Generation of Digital Terrain Models from Satellite Images by Stereo". *Acta Astronautica*, 15(3), 171-180, March 1987.

Cooper, Paul R. and Susan Hollbach. "Parallel Recognition of Objects Comprised of Pure Structure". Proceedings, DARPA Image Understanding Workshop, Los Angeles, Calif. February, 1987.

Feldman, Jerome A. "Four Frames Suffice: A provisionary model of vision and space." *The Brain and Behavioural Sciences 8*, 265-289, 1985.

Grimson, W. Eric. L. *From Images to Surfaces.* Cambridge, MA: MIT Press, 1981.

Grimson, W. Eric. L. "Computational Experiments with a Feature Based Stereo Algorithm", *IEEE Trans. on Patt. Anal. and Mach. Int.*, PAMI-7(1), 17-34, Jan. 1985.

Hannah, Marsha J. *Computer Matching of Areas in Stereo Images.* Stanford University AIM-239, Stanford, California, 1974.

Marr, D. *Vision.* San Francisco: W.H. Freeman and Company, 1982.

Mayhew, J.E.W. and J.P Frisby. "Computational and Psychological Studies Towards a Theory of Human Stereopsis", *Artificial Intelligence*, 17, 349-407, 1981.

Ohta, Y. and T. Kanade. "Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming". *IEEE Trans. on Patt. Anal. and Mach. Int.*, PAMI-7, 133-154, 1985.

Panton, Dale J. "A Flexible Approach to Digital Stereo Mapping", *Photogrammetric Engineering and Remote Sensing*, 44(12), 1499-1512, 1981.

Roberts, L.G. "Machine perception of three-dimensional solids" in *Optical and Electro-optical Information Processing*, J.P. Tippett et al. (Eds.). Cambridge, MA: MIT Press, 1965.

Yuille, A.L. and T. Poggio. "A Generalized Ordering Constraint for Stereo Correspondence", MIT AI Lab AIM-777, Massachusetts Institute of Technology, Cambridge, MA, May 1984.

Witkin, A.P. "Scale Space Filtering", in *From Pixels to Predicates,* Alex Pentland (ed.). Ablex Publishing Corp., 5-19, 1986.

Witkin, A.P., D. Terzopoulos and M. Kass. "Signal Matching though Scale Space". *International Journal of Computer Vision*, to appear.

Witkin, A.P. and J.M. Tenenbaum, "On the role of structure in vision", in *Human and Machine Vision*, J. Beck, and A. Rosenfeld, (Eds.). New York: Academic Press, 1983.

Parameterization of Mottle Textures

C.M. Brown, Elizabeth Hinkelman, Sanjay Jain
Department of Computer Science
University of Rochester

TR 217
June 1987

Abstract

Parameterization of textures can be useful for detection of textual similarities and matching. In this project we have developed a stochastic model to generate a set of parameters from the texture image domain and frequency domain. This model is aimed at quantification of textures for detection of similarities and differences . Our attention has been concentrated on the parameterization of mottled textures.

To test the model we have used it to generate texture images back from the parameters obtained from image analysis. The similarities and differences between the generated image and original images are used to refine and test the parametric model.

This work was supported in part by U.S. Army Engineering Topographic Laboratories research contract No. DACA76-85-C-0001, in part by an Eastman Kodak Company Grant/Fellowship and in part by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force base, New York 13441-5700, and the Air Force Office of Scientific Resarch, Bolling AFB, DC 20332 under Contract No.F30602-85-C-0008. This contract supports the Northeast Artificial Intelligence Consortium.

We thank the Xerox Corporation University Grants Program for providing equipment used in the preparation of this paper.

1 Introduction

The analysis of texture is often an important step in classifying and analyzing image fields. Yet texture measurement remains a highly nebulous subject. Many measures have been proposed, but their effectiveness is questionable because the structural characteristics of textural fields are still ill defined [8]. In a survey of the subject Haralick contended that "despite its importance and ubiquity in image data, a formal approach or precise definition of texture does not exist"[4]. Over the past few years various techniques have been developed for the analysis of textures. These techniques are basically of two types -

1) Structural methods

Textures can be described in terms of a set of primitives and placement rules. This point of view was first expounded by Rosenfield and Lipkin [10]. This sort of framework has similarity to language where the primitives can be seen as being symbols with the placement rule as the grammar for the language. The textures that can be described easily using this kind of framework have well defined primitives with obvious placement rules. For example in a chessboard the white and black squares can be viewed as the primitives with replacement rules to replace a white (black) square by a white (black) square with black (white) adjacent to it. The rules also ensure that the replacement rules can only terminate if the board is a square.

2) Statistical methods

Structural methods though preferable would usually be hard to obtain from a given image unless the broad characteristics of the image is known beforehand. The various methods of statistical analysis of images studied include the consideration of auto-correlation,Markov processes and co-occurrence statistics [8]. The statistical measures are specially attractive due to the ease with which they can be obtained from the image. The disadvantage of the statistical methods is that in practice different textures may give the same set of parameters. If enough parameters are used so that different images give different sets of parameters then the amount of parameters required can be huge. However the statistical methods are useful if only the broad characteristics of the image is desired. Moreover if the range of images on which the parameterization is to be done is known and we are to disambiguate among them then the parameters can be selected to suit the application.

2 Parameters

It is convenient to parameterize an image in some way so that similarities in images can be detected. The way the image characteristics vary over some domain can be a useful criterion for detecting similarities. With a constrained domain it may be possible to parameterize a texture model in a useful way. In our case the goal is to quantify the informal notion of the mottle severity and to do it we employ a statistical model.

2.1 Frequency Domain

The frequency domain parameters are useful in expressing the spatial regularities in the image. The radial density in the frequency domain indicates the size distribution of image components (for example the size of the square in the checkerboard). Many textures show preferential density at only a few frequencies depending on the number of basic texel types in the image. The angular density indicates the orientation of the components of the image. If the texels of the image follow certain specific placement rules then the orientation of the image determines the angular density. On the other hand if the orientation of the texels are not at all constrained by the placement of the neighbouring texels then the angular density is more or less uniform throughout the angular domain. The orientation distribution may characterize the image to certain extent. For example in a chess board image the angular density is concentrated in two perpendicular directions whereas an image of randomly oriented tiles shows uniform angular density.

2.2 Spatial Domain

Spatial domain parameters indicate the gray level distribution itself. For the textures that do not show selective frequency or angular preference the image domain parameters indicate how the image characteristics are distributed. The various statistical image domain parameters that can be considered useful include the mean, variance, skewness and kurtosis of the pixel gray level values. The parameters thus obtained are likely to be the same for similar images and different for dissimilar images.

2.3 Reconstruction

For texturesthat show regularity or controlled randomness in the frequency domain it may be possible to reconstruct the main characteristics of the images using the frequency domain parameters. Our model for reconstruction assumes such characteristics. The concentration has been on the frequency domain parameters because mottle is a large-scale phenomenon whose statistical characterization in the spatial domain seems impractical [5].

For regeneration of images from the frequency domain parameters we distribute a fixed number of points in the frequency domain using the parameters of radial and angular density obtained from the Fourier analysis of the image. This is done so that the frequency and angular properties of the original image are maintained in the generated image. This frequency domain reconstruction is used to generate the image back by using the inverse Fourier transform. In all that follows, "reconstruction" actually means generation of a texture that should be similar to the original according to the relevant quantitative and psychophysical criteria. The goal is to produce textures that are different instances of what a viewer would call "the same" texture.

3 A Mottle Model

Mottled textures arise in both industrial and in natural environments. An industrial mottle sample and is shown below (figure 1) [3]. In appearance, mottle is coarsely irregular or blotchy, without sharp intensity changes. Neither pixel statistics nor structural texture description methods seem well adapted to characterizing mottled textures [4,8]. Fractal dimension was considered, but would be a better model for turbulence. Fourier methods appear to offer the best hope,

although they have been superceded in many domains. Both of the traditional fourier domain approaches are inadequate, though: human inspection of the power spectrum is not quantitative, and reduction to radial or annular bins provides too much unstructured detail.

Our model is a stochastic, terse description of the spatial frequency content of a sample and its contrast. Our hopes are that

(1) the parameters of our model will decouple,

(2) the parameters will correspond in an intuitive way with texture characteristics,

(3) the parameters may be extracted from data samples, and

(4) a simple combination of at most two measures will correlate well with industrial standard metrics for mottle severity [3].

If we are successful, most of the model parameters will be constant for a particular application, and the remaining simplified model will be useful in practice. However, there is significant leeway for extending this model (through more complex and interdependent probability density functions, for example.)

Our strategy is to develop and understand a generative model and then work on extracting its parameters from real images.

Phenomenologically it seems that mottle samples could be represented by distributions of points in frequency space. Each frequency space point can be represented by ρ, its radial distance from the origin; θ, the angle of that radial vector and Z, the complex vector representing the weight of this Fourier component. These parameters should be sufficient to generate mottle, but one important component of texture seems to be its "contrast". In the model so far, contrast depends in a complex way on the joint properties of the other three parameters. Thus our strategy is to characterize the geometric parameters as simply and independently as possible and to add a "contrast parameter". This decoupling is highly desirable from many points of view.

Assuming the parameters are independent, an MxM mottled image may be generated by the following simple model.

(1) $P(\rho)$, the probability distribution function of ρ (ρ varies from 0 to M/2 or substantially less.) This parameter is related to spatial extent of mottle.

(2) $P(\theta)$, the pdf of θ (θ varies from 0 to 180 degrees.) This parameter is related to the directionality of mottle.

(3) P(Zreal), P(Zimaginary) the pdfs of frequency components. These parameters are related to contrast, brightness, and appearance of mottle. We will refer to the two distributions by the shorthand, P(Z).

(4) N: the number of frequency components to use in texture synthesis.

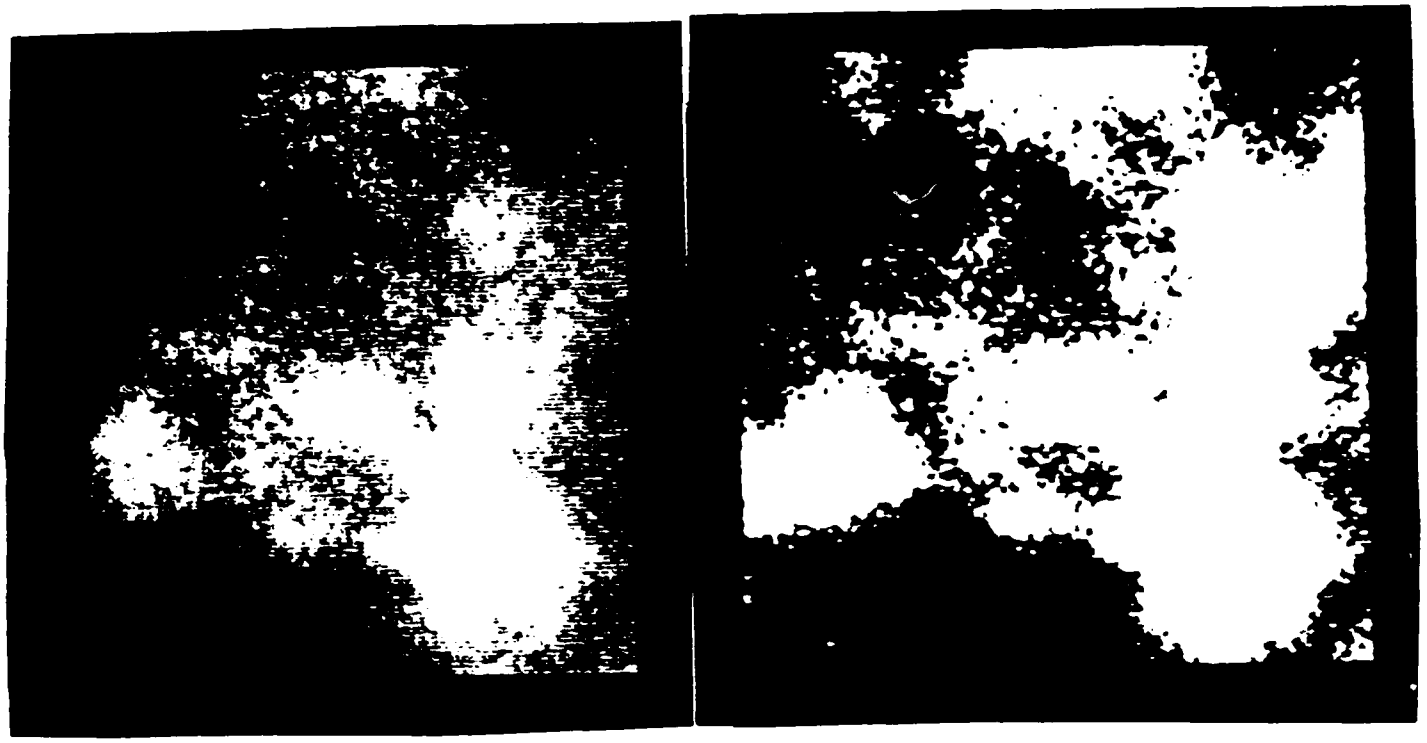


Figure 1: Mottle sample, courtesy Eastman Kodak Company. This image is a photograph of an Ikonas video display, showing a) digitized vidicon image trimmed to eliminate the effects of an unevenly lit light table, and b) the same image scaled linearly for maximum displayable dynamic range.

(5) S, a function mapping Image intensities X parameters --> image intensities, and acting as a scaling filter that provides variable contrast of the final output.

The process is to generate N random points (ρ, θ, Z) and place them along with their complex conjugate reflections in Fourier space, so that their inverse transform is real. $P(\rho)$, $P(\theta)$, and P(Z) are independent. Joint distributions are of course more general but harder to characterize, and we hope they are unnecessary. The inverse Fourier transform is applied, then S, the scaling function, is applied to adjust the "contrast" of the result. The computational steps in the model are diagrammed in fig. 2.

In our current version of the model we have chosen uniform distributions for the parameters ρ, θ, and Z, and (rather arbitrarily) an exponential $x' = x^Y$ scaling function. Another candidate is a linear ($x' = ax + b$) scaling model. The S function finally chosen should relate in a simple way to the underlying physical causes of mottle or to the phenomenal characteristics of industry standards. Each parameter is thus characterized by two probability distribution parameters. In the table below we give the probability distribution parameters that we have used in texture generation experiments. The mean and standard deviation are easily related both to the other distributions (especially gaussian and poisson) and to measurements on real data.

param	μ	σ	distribution
ρ	1-10	1/2-6	uniform
θ	90	52	uniform
Z	0	60	uniform
N	1-256	0	constant
γ	0.7-3.	0	constant

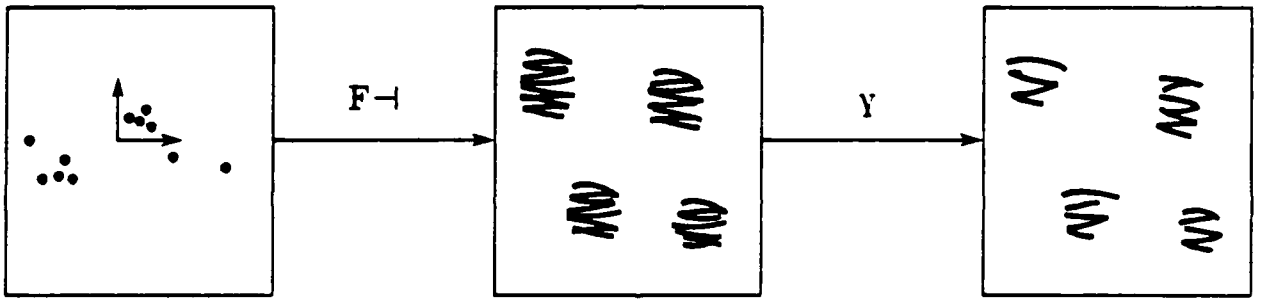


Figure 2: The series of transformations: From Fourier space to image space via an inverse Fourier transform, followed by gamma filtering.

4 Texture Generation Experiments

A program was written to generate textures according to the parameters given in the previous section. Representative samples of artificial textures are shown in figures 3, 4, 5, and 6.

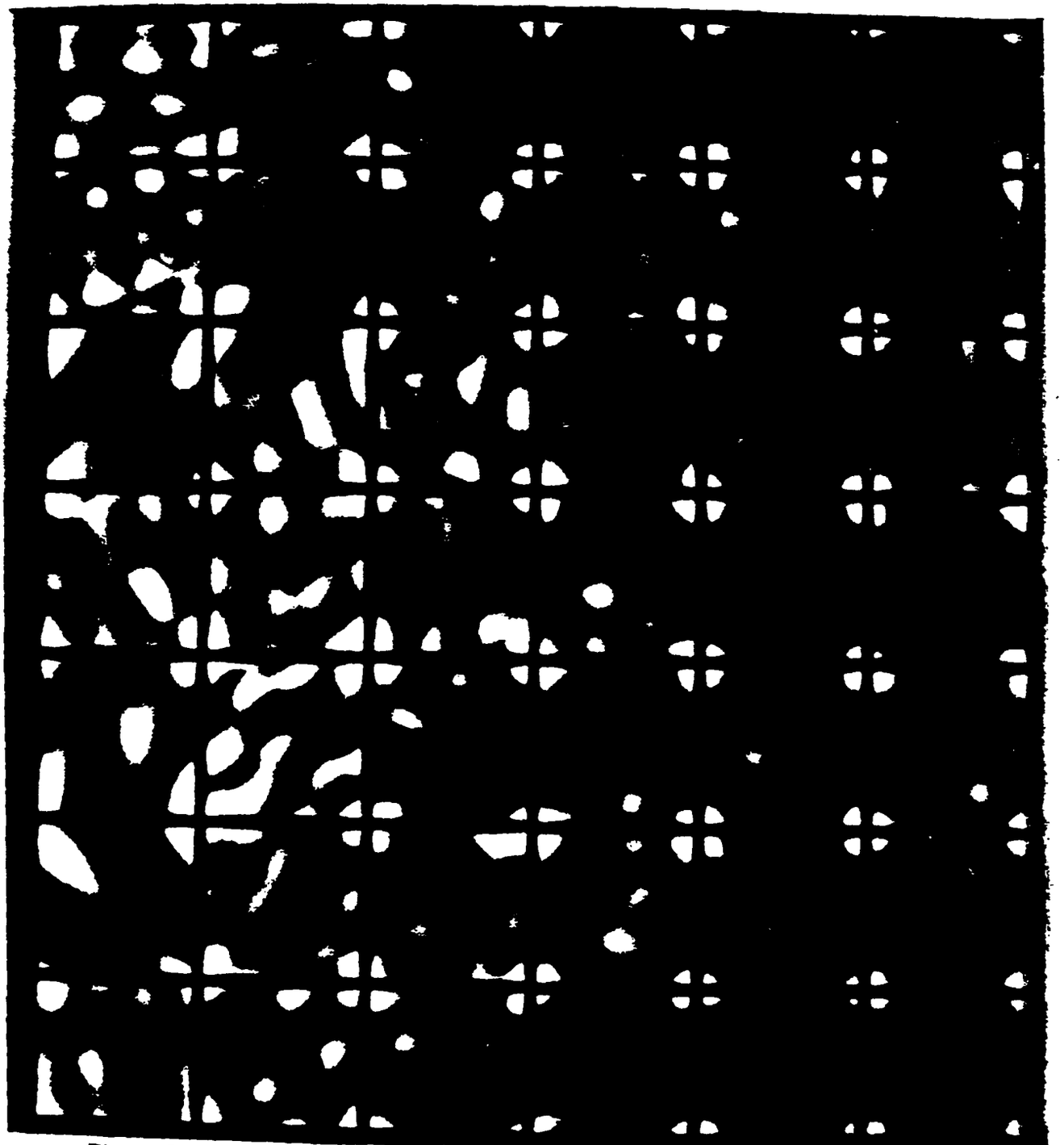


Figure 3: Artificially generated textures: $0 <= \rho <= 4$, $0 <= \theta <= 180$, Zreal = Zimaginary = 10, $\gamma = 0.9$. From left to right, the columns have N = 8, 16, 32, 64, 128, and 256.

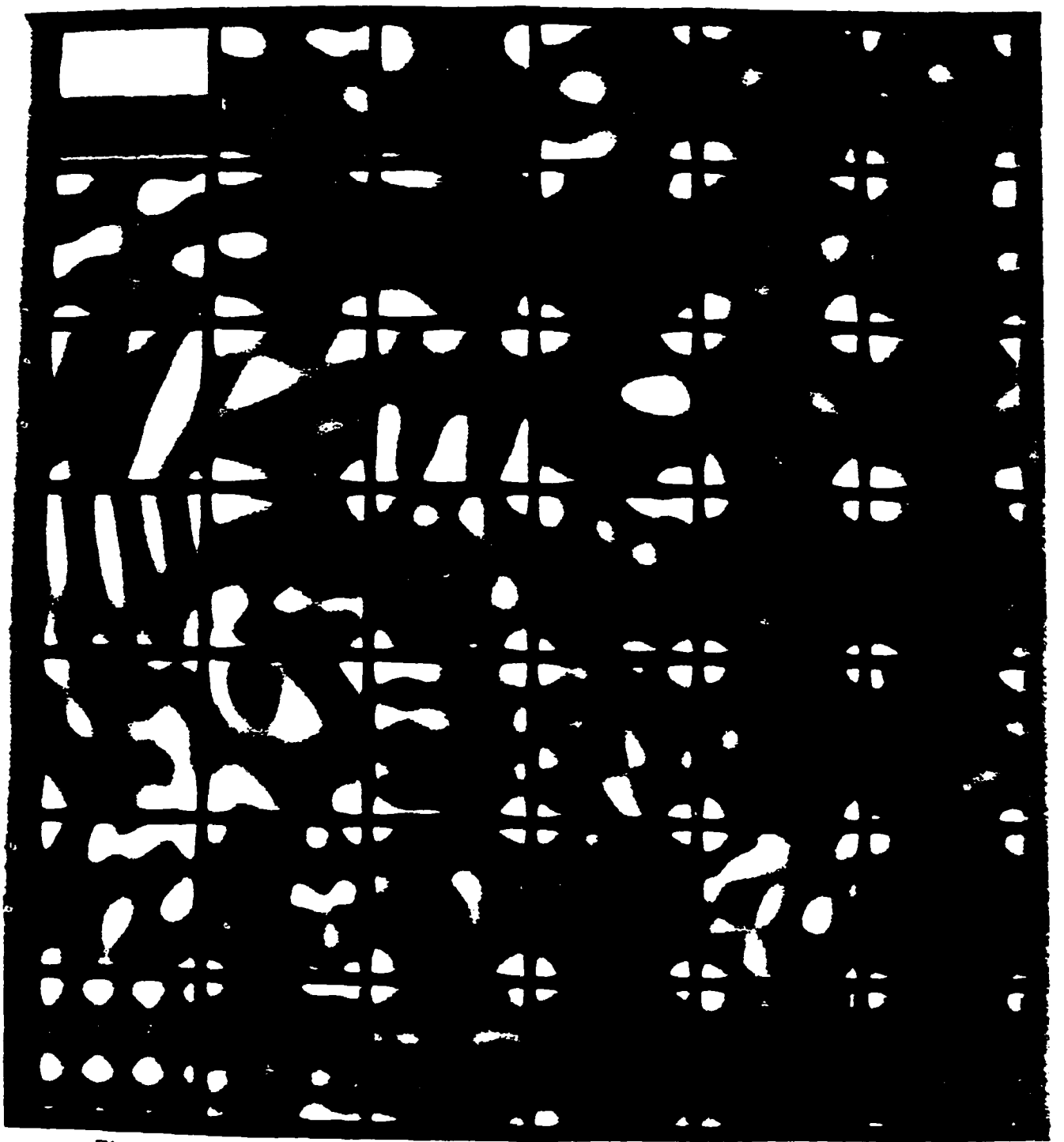


Figure 4: Artificially generated textures: $0 <= \rho <= 3$ for the first three rows and $0 <= \rho <= 4$ for the lower four, $0 <= \theta <= 180$, $0 <=$ Zreal,Zimaginary $<= 10$, ($\gamma = 0.9$). From left to right, the columns have $N = 8, 16, 32, 64, 128$, and 256.

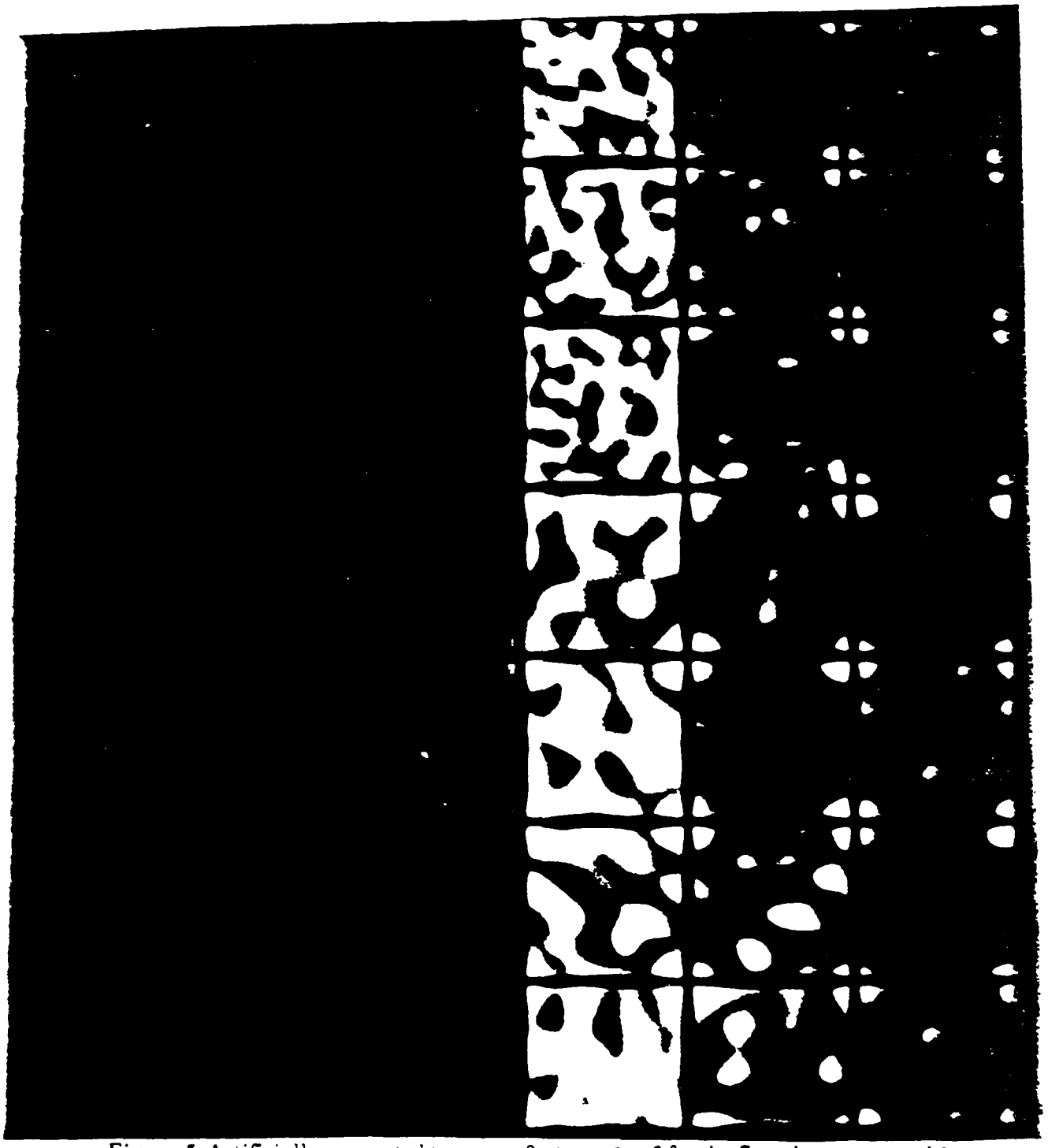


Figure 5: Artificially generated textures: $0 <= \rho <= 6$ for the first three rows and $0 <= \rho <= 4$ for the lower four, $0 <= \theta <= 180$, $0 <=$ Zreal,Zimaginary $<= 10$, $N = 64$. From left to right, the rows have $\gamma = 0.7, 0.8, 0.9, 1.0, 2.0$, and 3.0.

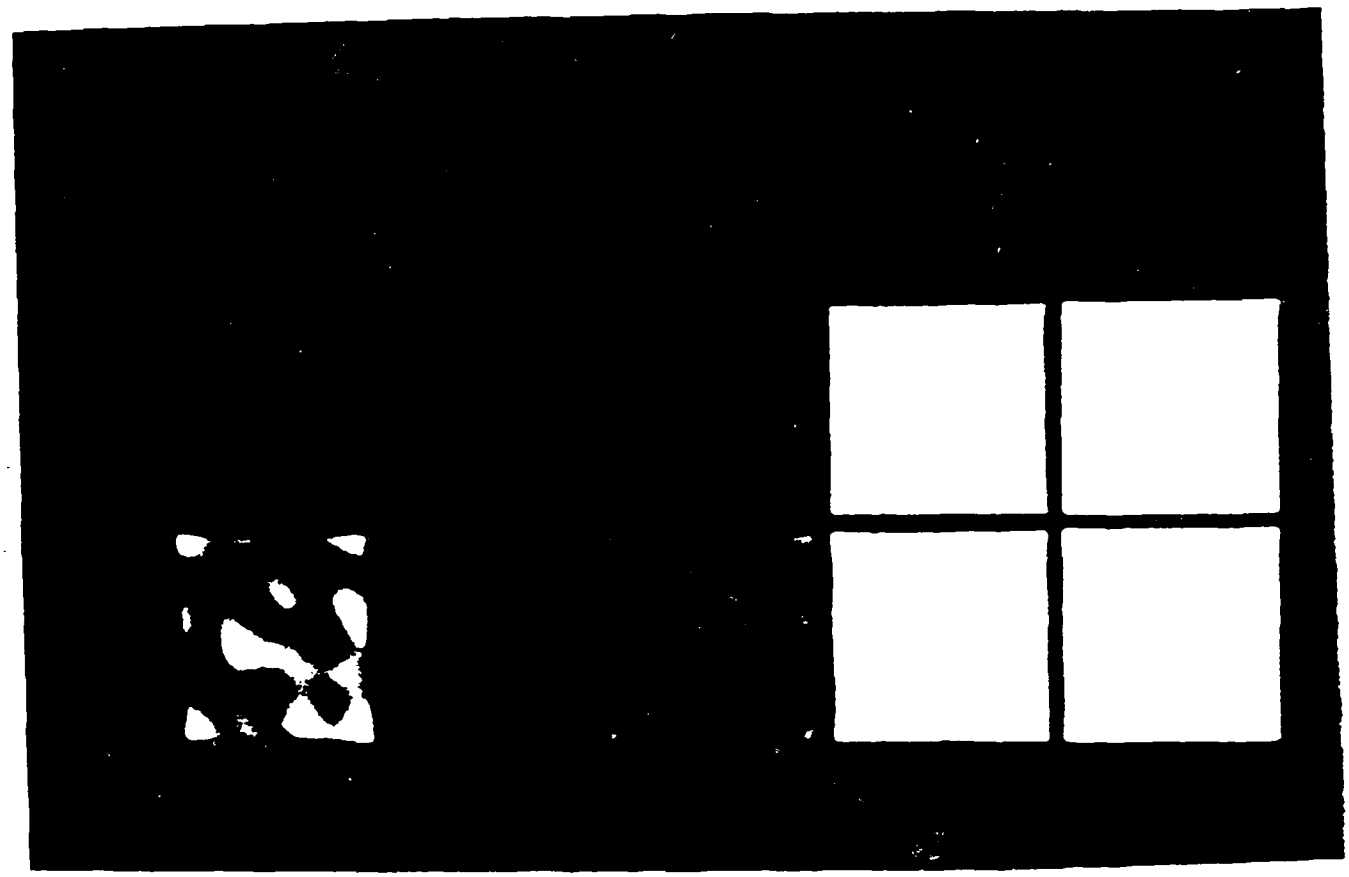


Figure 6: Artificially generated textures. The bottom left image is the original: the rest of the row shows the image compressed to 64 gray levels and displayed at offsets 0, 64, 128, and 192. The upper row uses the same offsets but only 32 gray levels.

5 Refinements

Generation of textures has verified that the fourier domain mottle model has some descriptive power. Computerized texture generation could also be a means to development of quantitative quality standards. A number of refinements could be made, however. These include:

(1) Multiplicative (linear) scale filter. This issue can be based on the physics of mottle formation or the psychophysics of human detection of mottle. We plan to treat it as an experimental issue so that we can simplify the relation of the scaling function to industrial standards.

(2) Normalized mean brightness. This parameter raises or lowers the overall brightness over which the mottle ranges. It can thus be used to ensure that the appearance of mottle reflects the differing effects of various film substrates. It is useful because it is psychophysically important: a dark image is hard to compare with a bright one.

(3) Addition of high frequency noise to simulate grain. This could be an independent parameter. It could alternatively be included by modifying the probability density functions $P(\theta)$, $P(\rho)$, and $P(Z)$.

(4) Joint probability distribution functions. If our variables cannot be adequately approximated by independent probability distribution functions, more sophisticated stochastic methods must be used.

(5) Poisson or Gaussian distributions. Uniform distributions will be useful, but other distributions could be even better.

6 The model parameters

For textures showing dominant directional or spatial frequency characteristics it should be possible to represent the sample using the radial distribution(denoting frequencies) and angular distribution (denoting angular direction preference) in the frequency domain. Moreover from the distribution of magnitude of the Fourier domain points we should be able to know how the concentration of angular and frequency preference is weighted.

To achieve our goal of texture quantification it is necessary to obtain some parameters from the image domain. The parameters that seem to be most useful are mean and contrast measurements with respect to the gray levels. Besides these it would be useful to know how the mean and contrasts vary when we look at different portions of the images. This quantifies certain basic variations between different portions of the image.

Thus the model consists of two type of parameters, one set in the frequency domain and another in the image domain.

6.1 Frequency domain parameters

The frequency domain parameters are useful for those images that show preferential spatial frequency or direction or both. The extraction of this type of characteristic is easier using the frequency domain directly. Imagine figuring out what the sizes of the texels are (given that the texels in the image have only a fixed number of different characteristic sizes) from the raw gray levels. On the other hand from the frequency domain the characteristic sizes may be found from the peaks in the radial spatial frequency distribution. Moreover if the image consists of many frequencies distributed in some random way (say normal, uniform or any such distribution) then studying the frequency domain radial density can give the parameters of the distribution easily. Moreover the distribution form can also be checked using various correlation techniques. The directional preference can also be easily found using the frequency domain. In such case we only have to look at the angular density. Uniform angular density indicates a random orientation of the texels whereas sharp peaks in the angular density histogram give the orientation of the underlying spatial frequencies and thus the texels themselves.

Thus the frequency domain parameters in the model consist of the distribution with respect to the radial and angular density. Besides this the distribution with respect to density at different magnitude of Z, Z_{real} and Z_{imag} in the frequency

domain was also considered. These distributions describing the frequency domain representation of the image are the following:

1) The distribution of spatial frequency density with respect to radial distance

This distribution can be calculated by summing up the frequency domain values of all the points at a given distance from the origin. If the probability distribution of the frequency is needed then it can be calculated by dividing the individual densities by the sum over the densities at all the frequencies. The formula for calculating the individual density is given by

$$P(k,\rho) = \sum_i \sum_j \left(\left| Z_k(i,j) \right| : Dist\left[(i,j), (X_c, Y_c) \right] = \rho \right)$$

where

$Z(i,j)$ is the power spectrum value at i,j

(Xc, Yc) is the center point

Dist. is the function to calculate distance

ρ is the radial distance.

k is for real, imaginary or magnitude.

In the above formula k is to denote the distribution with respect to real , imaginary or the magnitude of the Fourier tranform. Note that we should take the modulus for the real and imaginary distributions. The average frequency (radial distance) can be calculated by summing the product of radial distance and probability for that radial distance. Similarly the variance can be calculated by taking the product of the probability and the square of the difference of the radial distance and the average radial distance. Note that in a similar way we can calculate skewness, kurtosis and higher order terms by taking different powers of the difference from the mean.

2) The distribution with respect to angle

The distribution of the angular density can be calculated in the same way by summing up the Fourier domain values of all points such that the angle formed by the line joining the point to the origin with the X axis is θ. The probability distribution with respect to angle can also be calculated in a similar way by dividing the density by the sum of the densities at all the angles. The formula to calculate the angular density is given by

$$P(k,\theta) = \sum_i \sum_j \left| (Z_k(i,j) \right| : angle\left[(i,j), (X_c, Y_c) \right] = \theta$$

where

angle is the function to calculate angle.

θ is the angle

k is for real, imaginary or magnitude.

The average and variance of the angle can also be calculated in the same way as in 1. Again higher order terms can also be easily obtained from the probability distribution.

3) The distribution of Z the power spectrum value

The distribution of the power spectrum value can be calculated by calculating the number of points that have that power spectrum value. The probability of a particular point having a particular value of Z (the power spectrum value) can be calculated by dividing the density at Z by the total number of points. The average value of Z can be calculated by summing the product of Z and the probability of a point having Z as the power spectrum value. The variance can similarly be calculated by taking the sum over the product of the probability and the square of the difference from the mean. Again we can calculate the higher order terms also if required. The formula for calculating the density is given below.

$$P(Z) = \sum_{i} \sum_{j} (1) \Big| \; (Z(i,j) = Z))$$

where

$Z(i,j)$ is the power spectrum value.

4) The distribution of real part of the Fourier transform

The distribution of density at the different values of real part of power spectrum can also be calculated in the same way as 3 above. Note however that the magnitude of the real part is considered. Again the average and variance can be calculated in the same way as above.

$$P(real) = \sum_{i} \sum_{j} (1) : \Big| \; Real(Z(i,j)) \Big| \; = real)$$

where

$Real(Z(i,j))$ is the real part of Fourier transform

5) The distribution of the imaginary part of the Fourier transform

The distribution of density at different values of imaginary part of the power spectrum can also be calculated in the same way as 3 and 4 above. Note that in this case also we take the magnitude of the imaginary part of the power spectrum (otherwise the sum would be 0!).The average and variance can also be calculated similarly.

$$P(imag) = \sum_{i} \sum_{j} (1) : \Big| \; Imag(Z(i,j)) \Big| \; = imag)$$

where

$Imag(Z(i,j))$ is the imaginary part of Fourier transform

In representing the distributions, a conservative approach is to keep the complete distribution. However for characterization of the image the simple statistical parameters such as average and variance of the distribution should be enough since most textures show highly preferential frequency and direction preference. If the distribution of the various values (such as frequency and direction) is expected to be non uniform on the two sides of the mean then the higher order terms can also be used. In general, the adequacy of characterization of a particular domain by a particular set of parameters is an empirical issue.

6.2 Spatial domain parameters

The frequency domain parameters are often not enough to characterize the image. This is especially so if the image does not show uniformity in the frequency distribution. In such cases it is useful to go back to the image domain parameterization. Moreover the actual image domain parameters are more likely to indicate the form of images to humans. The distribution of grey levels in the local neighbourhood of the points gives an indication of what the image looks like. The overall distribution of the parameters obtained at local neighbourhood of the image indicates the global characteristics of the image.

The image domain parameters consist of the parameters obtained from the calculation of mean, variance and ratio of dynamic range to the maximum in the local neighbourhood of the image point varying over the image domain.

The mean denotes the average intensity at the local neighbourhood. The variance and the ratio of dynamic range and maximum indicate the contrast at the local neighbourhood of the point.

The overall mean and variance of the above desired distributions indicates certain global properties. For example the variance of variance indicates how the contrast varies as we move to different positions in the image.

1) The distribution of mean

The mean at the local neighbourhood of a point I(i,j) indicates the average intensity around that point of the image. We can consider the local neighbourhood as the points which are within a certain distance of (i,j) in some convenient metric. Any of the local neighbourhood criteria can be used. We have used the city block metric in our work. The formula for calculating the average intensity at a point (i,j) can be given as follows

$$\mu(i,j) = \left(\frac{1}{m^2}\right)\left[\sum_x \sum_y I(x,y)\right]$$

where

x and y vary over the local neighbourhood.

m is the size of the neighbourhood.

$I(x,y)$ is the gray level value at the point (x,y)

The mean of the above μ can be calculated by summing up the μ 's and dividing by the total number of points for which μ was taken. The variance can also be calculated as it was calculated for the frequency domain parameters. The mean of mean gives the indication of global mean. The variance of the means is one measure of perceived contrast , "but contrast is a complex perceptual phenomenon".

2) The distribution of the variance

In some models of texture, variance in the local neighbourhood indicates the local turbulence in the grey level values. Moreover the variance in the local neighbourhood can also be used to determine if the size of the neighbourhood chosen is reasonable for the given image. The desirable size of the neighbourhood should not be much larger than the texel size. The size of the neighbourhood should similarly not be too small. The variance at different neighbourhood size can thus give indication of the texel size. The variance of a local neighbourhood of the point is calculated as follows

$$v(i,j) = \sigma^2(i,j) = \left(\frac{1}{m^2}\right) \sum_x \sum_y \left[I(x,y) - \mu(i,j)\right]^2$$

where x,y vary as above

The mean and variance of the variance can be calculated in the same way as other means and variances. The mean of the variance indicates the average turbulence in the gray levels. The variance of variance would indicate how the contrast varies over the whole image.

3) The distribution of dynamic range-maximum ratio

The dynamic range of the gray level is defined as the difference between the maximum and the minimum values of the gray levels.

The ratio of the dynamic range and the maximum gray level value in the local neighbourhood of the point is another form of contrast measurement which indicates how steeply the gray level value is changing in a local neighbourhood. Again this can also be used as the guide for guessing the texel size. The formula to obtain this ratio is

$$R(i,j) = \frac{max\,(I(x,y)) - min\,(I(x,y))}{max\,(I(x,y))}$$

where x,y vary as above.

The mean and variances of R(i,j) are calculated in the same way as we calculated the means and variances of other parameters. The mean of this ratio indicates the expected local steepness in the gray level values. The variance indicates how the steepness varies over the whole image domain.

Instead of the actual distribution, the number of parameters can be reduced by keeping only mean and variance of the above three distributions. The mean and variance would indicate the global average and variation in the local properties. These parameters can be used to quantify certain aspects of the image in spatial domain.

7 Experiments

We ran several experiments for three types of textures.

1) A digital image of a mottle standard from Eastman Kodak Company.

2) Unfocussed images of certain textures from Brodatz's book[2].

3) synthetic mottle generated by running the model in a generative mode (see Figs. 3-5) [4].

Tables 1 and 2a&b show parameters derived from these textures and demonstrate a certain consistency with intuition about textural similarity. Figs. 7 and 8 show representative textures, a subset of those used in the study.

Image	Avg μ	Avg V	Avg R	Var μ	Var V	Var R
mot2	122	25.9	0.19	173	257	0.002
mot1	132	16.7	0.16	23	33.1	0.0005
Br4	173	258	0.50	79	9987	0.012
Br1	239	580	0.49	49	1.76E5	0.022
Br3	166	2613	0.85	151	3.3E5	0.005
Br5	135	2589	0.93	130	2.8E5	0.002
Br2	230	1176	0.59	49	1.65E5	0.01
Hin2	124.64	722.36	0.577	332.07	2.63E5	0.011
Hin1	125.18	604.4	0.51	331.58	3.76E5	0.0267
mot3	222.98	994	0.46	290.41	2.52E5	0.009

Table 1: Spatial domain data

Image	Var ρ	Var θ	Var Z	Var Z_r	Var Z_i
mot2	595	3.328	4.16 E6	9.78 E5	3.2 E6
mot1	524	3.265	4.84 E5	2.01 E5	3.17 E5
Br4	288	3.24	4.12 E6	1.62 E6	2.82 E6
Br1	301	3.26	7.19 E6	3.73 E6	4.00 E6
Br3	320	3.20	2.84 E7	1.58 E7	1.58 E7
Br5	309	3.21	2.89 E7	1.54 E7	1.67 E7
Br2	269	3.17	1.49 E7	7.29 E6	8.63 E6
Hin2	367	3.21	3.13E6	1.64E6	1.48E6
Hin1	380	3.45	2.95E6	1.18E6	1.78E6
mot3	390	3.45	1.87E7	9.34E6	9.91E6

Table 2a: Frequency domain data

Image	Avg ρ	Avg θ	Avg Z	Avg Z r	Avg Z_i
mot2	38.41	3.02	559.77	336.46	379.45
mot1	43.65	3.11	421.73	264.54	270.60
Br4	31.37	3.09	1284.4	800.51	828
Br1	29.26	3.11	1713	1095	1091
Br3	33.40	3.09	4157	2686	2619
Br5	32.60	3.07	4065	2542	2633
Br2	29.07	3.04	2350	1460	1529
Hin2	13.11	2.73	86.02	57.88	55.62
Hin1	13.37	2.94	82.87	47.30	59.53
mot3	25.87	3.1	1713	1099	1086

Table 2b: Frequency domain data

8 Reconstruction

As observed before for textures showing special characteristics in the Fourier domain it should be possible to reconstruct an image similar to the original from the

parameters calculated in the frequency domain. For textures such as mottles it may also be possible to reconstruct using only the image domain parameters. For reconstruction from spatial domain parameters we can give the gray level values of the pixels according the the distribution in the neighbourhood if the distribution of the various parameters in the local neighbourhood of the point is kept. For textures that seem more or less uniform we can distribute the gray levels based only on the global characteristics.

For reconstruction from the frequency domain parameters we can distribute some points in the frequency domain according to the density in the radial and angular bins. For textures that show very concentrated radial and angular preference only the statistical parameters of the radial and angular distribution may be needed for generation of points.

8.1 Reconstruction from frequency domain parameters

For reconstruction it might be expected that we would require to keep a large number of parameters. However if the texture shows a special form of distribution in the radial and angular direction then by knowing the characteristics of the distribution we can generate the frequency domain points according to the characteristics of the distribution. For example suppose we know that the frequency domain points of the image show a uniform distribution with certain mean and variance for both radial and angular direction. Then we can distribute the points uniformly in the frequency domain with the corresponding mean and variances. Various mottles were generated using such distribution with certain modifications. The resemblance of the synthetic images with the actual mottles gave the motivation for this project. We can similarly generate textures for various kinds of distribution. If the distribution selected is one that is obtained from the parameters(or the whole distribution) obtained from the actual image we should get an image similar to the original image. In our experiments so far we have used the full distribution of ρ, θ and real and imaginary Z distributions for generation, rather than reconstructing the distributions from a few parameters.

The reconstruction from the distribution can be done as follows:

1) Select number of points N that will be distributed in the frequency domain .

2) Generate N points in the frequency domain according to the distribution (i.e. generate X,Y,Zr,Zi for each point such that the distribution and density of the points follow the characteristics of the original distribution).

3) Take the inverse Fourier tranform to generate the image.

4) Scale the image gray levels appropriately.

The number of points selected in the Fourier domain should not be too small because in that case the distribution may not be able to capture the characteristics of the image. Step number 2 above can be performed in various ways since distributions are approximated by histograms and, in practice we can get similar histograms by distributing the points and weights at those points in a number of ways. This step is discussed in detail below. The scaling of the image is required since the inverse Fourier transform thus generated may have negative values (since the generated frequency domain points do not necessarily give an exact Fourier transform of an image with positive gray level values). Moreover since the average

intensity of the points may change we would have to do appropriate scaling of the image so that the image generated would have similar intensity to the original (since the frequency domain analysis is supposed to give only the frequency and angular distribution). This step can be done by calculating the average intensity of the original image.

8.2 Distributing points according to desired histograms

The distribution of points according to the model or derived (approximate) distribution parameters can be done in several ways. Some of the methods are discussed below.

When the whole distribution of the radial and angular density is kept:

a) We can just distribute the points according to the radial and angular probability distribution in the frequency domain. For this we can keep a cumulative distribution of probability of any radius or angle. A random value (uniform) is generated according to any of the standard random number generating functions. The radius (angle) in which this value falls in the cumulative distribution can be taken as the radius (angle) of the generated point. The magnitude of the real and imaginary part of the power spectrum can be chosen randomly according to their distribution. This is the simplest way to distribute the points. The sign of the values can again be taken randomly with probability half. Note that the points generated should have a complex conjugate partner at a position (-x,-y) (x,y is the position of the generated point) . This is done so that we can get a real image after taking the inverse transform.

b) Since a single high valued point at a higher frequency may generate high frequency components even though the sum of the values at that frequency is not considerable we would like to reduce the magnitude of the values at the points at high frequency. This can be done in two ways. The simplest way would be to reduce the intensity at any distance R by dividing it by R^2 (i.e proportional to the number of points at that distance). This method discourages the high frequency components, Alternatively we can blur the point at high frequency. This reduces the magnitude of points at high frequency while keeping the total sum of values at high frequency according to the distribution. Note that we would have to do superposition if the effect of more than one point blurring is felt at some other point. This blurring can be done in any form of distribution (gaussian or other template such as 1/2 at center and rest in the neighbourhood of size depending on the frequency). Other values can be found in the same way as in (a) above.

c) In this case we can distribute the points randomly but take the weight at those points according to the distribution. The points can be generated according to any uniform distribution. In this case we would not have the problem of high frequency having concentrated values. The total sum at any frequency follows the distribution since there will be more points generated at the high frequency. Note that the value generated at a frequency should have the dividing factor of R^2.

d) Another variation would be to combine the two ideas above i.e to generate the points according to the probability distribution and then generate the

values according to the density at those points. This method would disfavour the high frequency components. This might be useful when high frequency components are to be suppressed.

In all the above methods we keep the full distribution of the probability density at any radius or angle. The strategy changes slightly when we have to generate according to the distribution of points that follow a distribution pattern such as normal, uniform or any known such distribution. Using distribution parameters would be useful in decreasing the amount of information required to characterize the texture. Not much work has been done on this kind of generation since the results using distribution information are still not fully understood. Work in the immediate future will concentrate on the regeneration issue. One idea is to distribute the points and density in much the same way as above but instead of using the uniform generation or the generation according to the distribution of density at a frequency and angle we would generate points and values at the points using a random generator which follows the characteristics of the known distribution. Random generators for non-uniform distribution are more complicated (Knuth discuses the generation of random numbers according to various types of distribution) [7]. Alternatively the characteristic distribution function of the desired distribution can be computed analytically and used with the existing software.

8.3 Reconstruction from image domain parameters

For images that appear random and only the contrast or mean intensity seems significant it may be possible to reconstruct the texture from the image domain parameters. Moreover certain textures may not have a well defined frequency preference nor have a reasonable pattern in which the frequencies are distributed. For such images the frequency domain reconstruction may not give good reconstruction. Thus it may be desirable to study the techniques to reconstruct the images from the image domain parameters. The reasons for this type of reconstruction are similar to the choice of using the image domain parameters. Some mottled textures seem to be a likely candidate for this type of reconstruction. However not much thought has been given to image domain models. The procedure followed to reconstruct the image are quite similar to the frequency domain reconstruction case.

In this case we take the parameters on the image domain. Then using the distribution we calculate the value of the gray level at each point to obtain the reconstructed image. If the statistical characteristics (such as mean, variance, dynamic range) are kept for a neighbourhood of points selected at appropriate distances (the appropriate distance being the size of the neighbourhood) then we can generate the gray level values at the local neighbourhood according to the statistical parameters at that distribution.The procedure for generating the gray level values is the following:

For each selected point for which the local parameters were kept:

> generate the gray level values of all the points in the local neighbourhood of the point such that it satisfies the mean, variance in that neighbourhood. This can be done by assuming normal or uniform distribution in the local neighbourhood. The resulting values can be appropriately scaled to satisfy the dynamic range constraint.

Note that due to normal or uniform distribution of gray level values in the local neighbourhood the size of the neighbourhood should not be large (as compared to the texel size). Moreover if we take the size to be too small then the number of parameters kept would increase as the inverse of the square of the neighbourhood size.

If the texture doesn't show any particular changes in mean,variance or dynamic range as we concentrate on different portions of the image then we can keep only the overall characteristics of the local characteristics. The local characteristics can then be obtained by randomly selecting the local characteristics according to the global values. The regeneration can then follow in the same way as if we had the local characteristics.

9 Experiments on regeneration

Most of our work on regeneration of images was on the regeneration from frequency domain parameters. We ran our model for reconstruction on various mottled images from the Brodatz's book, a checkers board image and some of the synthetic mottles of Hinkelman. Some of the images on which the model was tried along with their power spectrum, the generated power spectrum and the regenerated image are shown in Figs. 9-13. The schemes used for the regeneration were the following:

Fig. 9 (checkerboard image) -- schemes a and c mentioned in section 8.2 .

Fig. 10 and 11 (Hin1 and Hin2) -- schemes a and b mentioned in section 8.2.

Fig. 12 and 13 (Br3 and Br) -- schemes b and c mentioned in section 8.2.

The results were good for the checkerboard image. The results were not so good for the mottles. The power spectrum of the various images were similar to the generated ones as expected. The results for the images obtained by taking the inverse Fourier transform were not as good as expected in the case of the mottles, even though in most cases some of the characteristics of the images were captured in the generated images.

10 Discussion

The parameters generated showed good similarities for similar type of images. The method thus shows some promise for parameterization of the mottles. The reconstruction for images which show a high degree of regularity such as checkerboard can also be done with some correspondence between the original and reconstructed image. The model thus could be used for reconstruction of textures showing high degree of regularity. The reconstruction of mottles from the parameters was not as successful. This may be due to the reason that there is a lesser degree of regularity in the mottled images. To reconstruct the mottles we can try variations of the idea so that the the generated images resemble the original image to a greater extent. Various modifications of the above ideas can be tried so that a better fit for the mottles can be found. Future work will concentrate along these lines. The various models that can be tried are

1) Use the image domain parameters in the regeneration of the image.

2) Use the Fourier domain parameterization to reconstruct the image. Use the image domain parameters on the generated image to force some of the image domain characteristics on the image.

3) Use various contrast measures obtained from the image (the contrast measures can be the dynamic range, the variance etc.) and force them on the regenerated image. This is likely to ensure that the various contrast measures are preserved. Note that in forcing one of the characteristics we may violate other properties. The way would be to find a compromise between the various characteristics.

Another important topic is to extract the "contrast" parameter (the arguments to the function S). This is much more interesting, especially because it is our hope that the $P(\rho)$, $P(\theta)$, and $P(Z)$ can be "frozen in" for a situation and the contrast parameters will provide the quantitative measure of mottle severity. In this case one way to extract the contrast parameters would be to choose a likely value (say $\gamma 0$ for definiteness), "unscale" by $\gamma 0$, (ie apply the inverse of the scaling using $\gamma 0$), then do steps 1, 2, 3 above and check to see the resulting (μ, σ)s fit the mottle shape model. If they do not, change $\gamma 0$ to $\gamma 1$ and repeat. The hope is that a hill-climbing process on γ will find that γ which results in the best fit of the other mottle parameters to the data.

An alternative approach to quantitative analysis with the model does not assume that the $P(\rho)$, $P(\theta)$, and $P(Z)$ can be "frozen in". In other words, it allows these parameters to be calculated as part of the image quality estimation. Rather than expending large amounts of effort on calculating a very accurate value for γ based on the other parameters, we fall back on our independence assumption. The first two steps of the analysis process would be as in verifying the model, but rather than finding γ by means of a complicated iteration (hill-climbing), use some simpler method:

(1) Fourier transform the image,

(2) Compute μ, σ of $P(\rho)$, $P(\theta)$, and $P(Z)$

(3) Apply a simple contrast measure.

In this case, the hope is that some carefully chosen but easily computable contrast measure will be adequate to distinguish relevant variations in images. Such a metric could address some of the psychobiological issues mentioned below, or it could be a simple rule of thumb that happens to be effective. The difference between the maximum and minimum image values is a very primitive measure: another is the variance of the gray levels. The Spatial Gray Level Dependence contrast measure is somewhat more sophisticated, and presupposes some estimate of the scale. None of these has much psychophysical value, though. It is worth noting that the limiting factor in performing the above analysis on line is the size of the area analyzed, which determines the time of the Fourier Transform. The approach mentioned above performs several Fourier Transforms and thus requires more time.

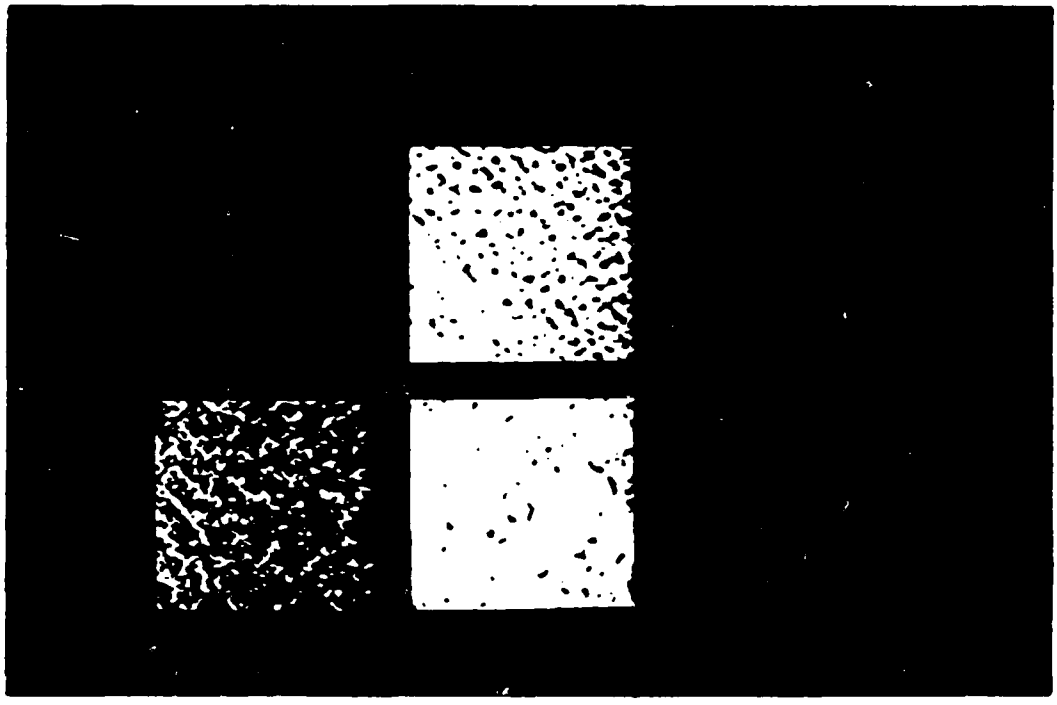


Fig. 7: Textures used in the experiment

upper left: mottle picture obtained from Kodak (mot1).
lower left: mottle picture obtained from Kodak (mot2).
upper middle: Brodatz's texture "Pressed cork" unfocussed (Br1).
lower middle: Brodatz's texture "Handmade paper" unfocussed (Br2).
upper right: Brodatz's texture "Pigskin" unfocussed (Br3)
lower right: Brodatz's texture "Pressed cork" unfocussed (Br4)
(different scaling than Br1).

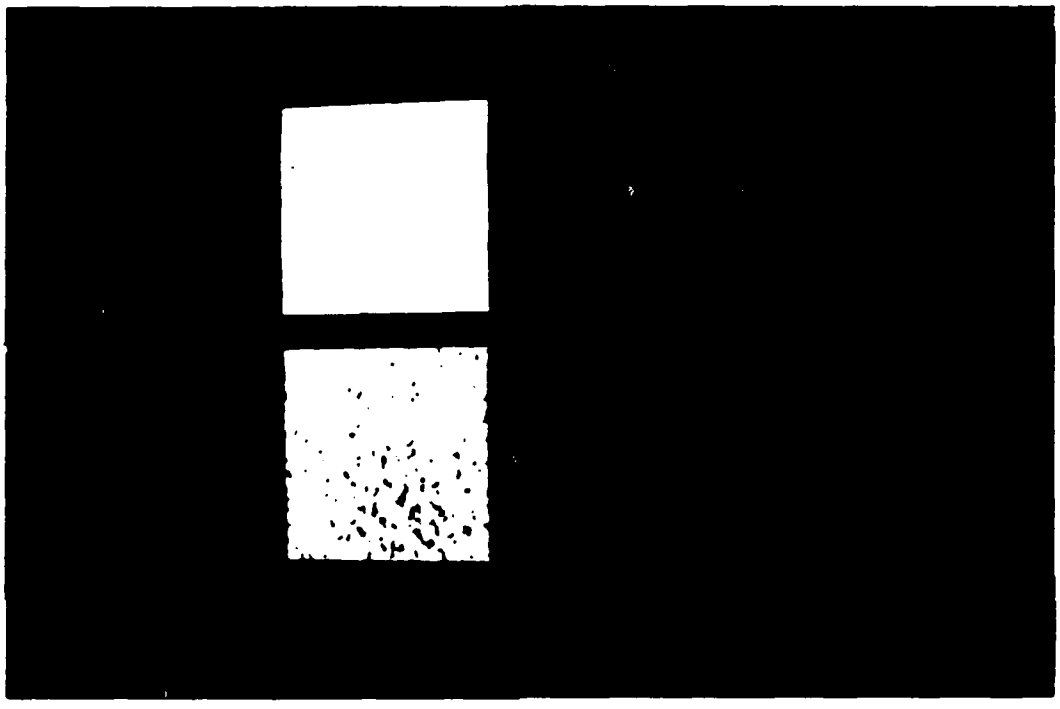


Fig. 8: Textures used in the experiment

upper left: another mottle texture (mot3).
lower left: Brodatz's texture "Pigskin" unfocussed (Br5) (different portion of the same image as Br3).
upper right: Generated texture 1 (Hin1).
lower right: Generated texture 2 (Hin2).

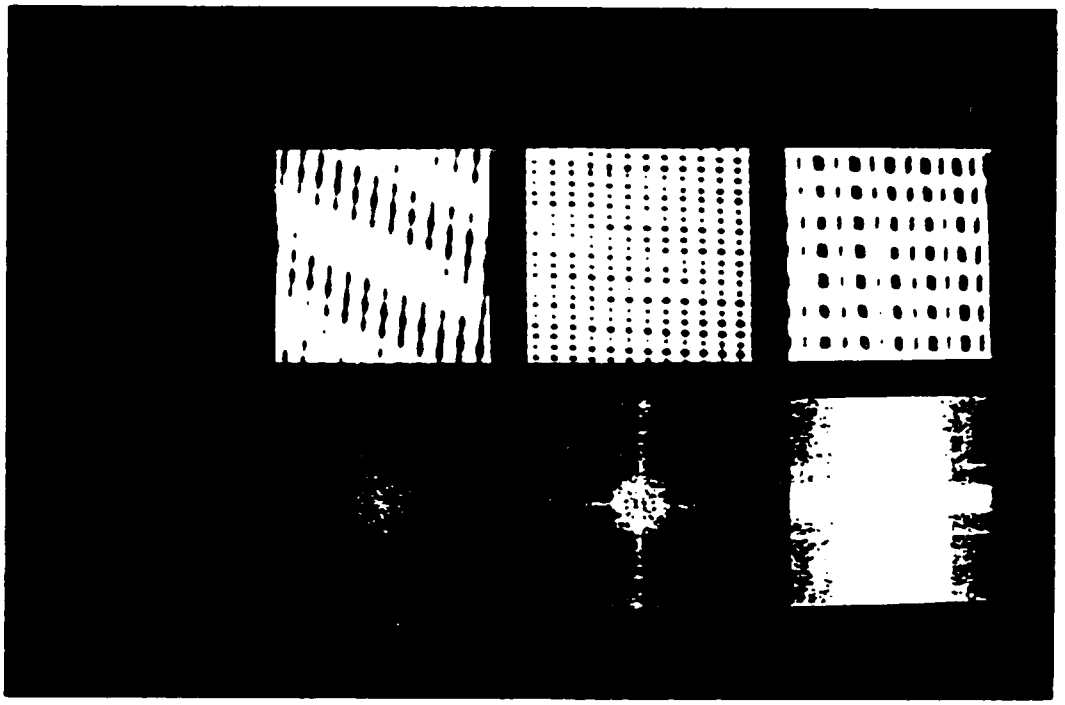


Fig. 9: Texture Reconstruction

lower left: The original image (chekerboard from [2]).
upper left: The power spectrum of the original image.
lower middle and right: Reconstructed images using different generation schemes (see text).
upper middle and right: Power spectrum of the reconstructed images.

Comment: In reconstruction 1 (middle) The frequency has increased in vertical direction. In reconstruction 2 (right) there is a tilt in the direction and increase in the vertical frequency. In reconstruction 1 the sharp parallel horizontal lines in the fourier transform have not been captured. In 2 due to the scheme chosen the lines have not been captured. Overall high amount of directional and frequency information has been captured.

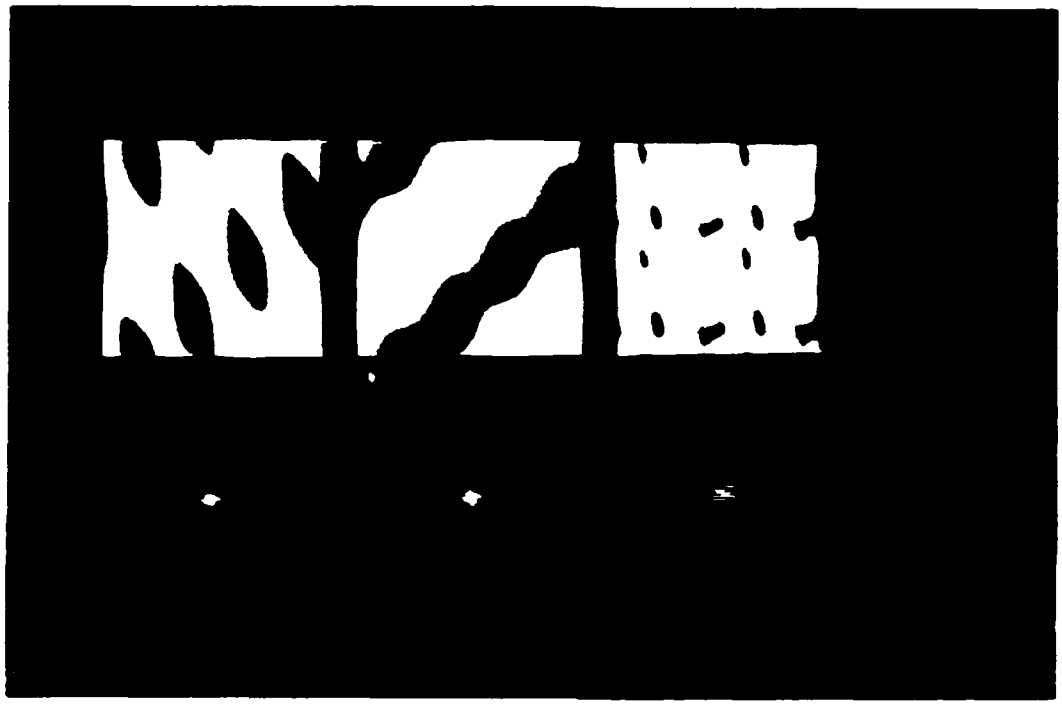


Fig. 10: Texture Reconstruction.

lower left: The original image (Hin1).
upper left: The power spectrum of the original image.
lower middle and right: Reconstructed images using different generation schemes (see text).
upper middle and right: Power spectrum of the reconstructed images.

Comment: The power spectrum in the reconstruction resembles to a high extent the original power spectrum. The slight amount of high frequencies in the original spectrum has not been captured satisfactorily. The net frequency in the image seems to have been reduced.

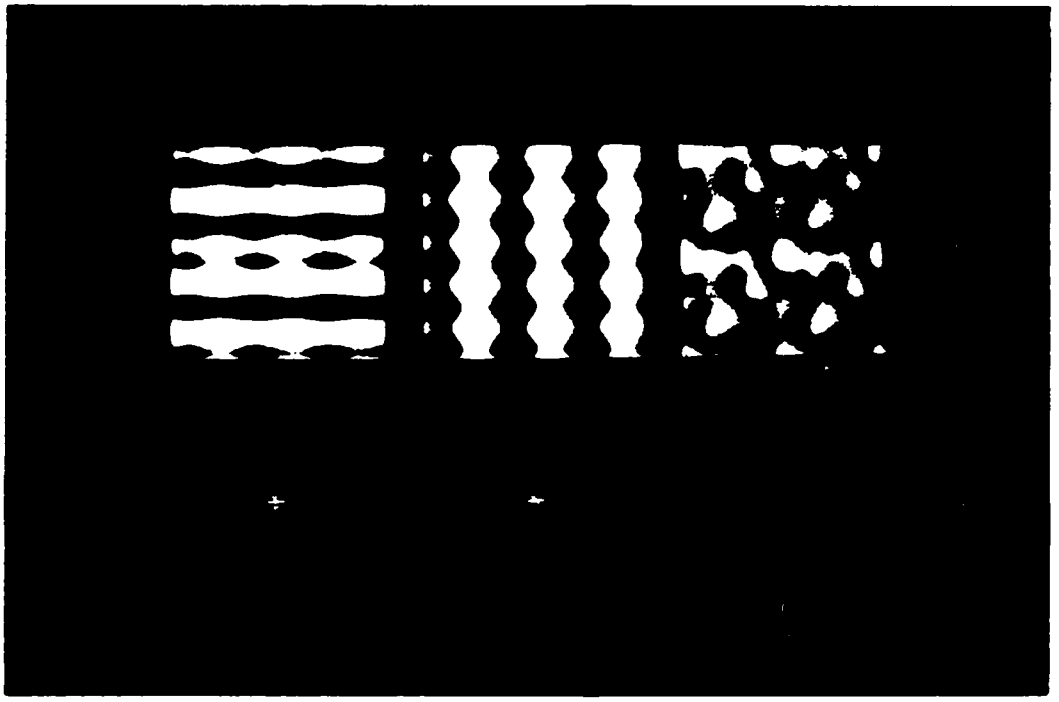


Fig. 11: Texture Reconstruction.

lower left: The original image (Hin2).
upper left: The power spectrum of the original image.
lower middle and right: Reconstructed images using different generation schemes (see text).
upper middle and right: Power spectrum of the reconstructed images.

Comment: The power spectrum in the reconstruction resembles to a high extent the original power spectrum. The slight amount of high frequencies in the original spectrum has not been captured satisfactorily. However the frequency component is better than in the previous (Hin1) image. The generated image has high angular preference which is missing in the original image.

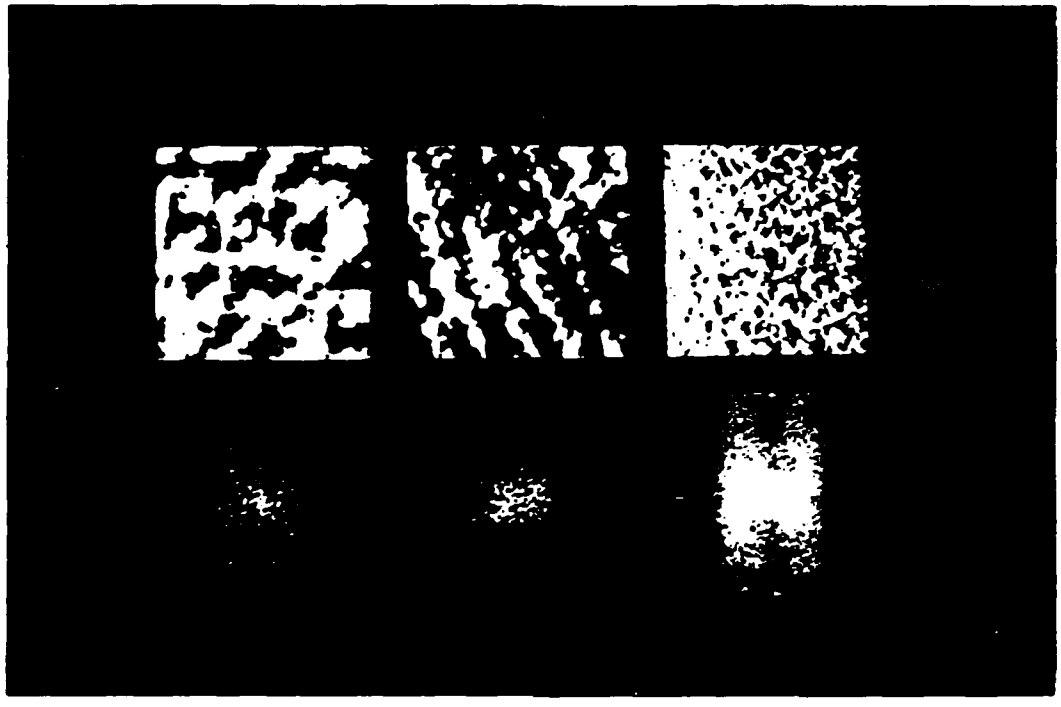


Fig. 12: Texture Reconstruction.

lower left: The original image (Br3).
upper left: The power spectrum of the original image.
lower middle and right: Reconstructed images using different generation schemes (see text).
upper middle and right: Power spectrum of the reconstructed images.

Comment: In this regeneration the brightness in the upper and lower middle portion of the power spectrum is missing in the reconstructed power spectrum. The image shows similarity in the frequency and randomness of the direction. The contrast is higher in the generated image.

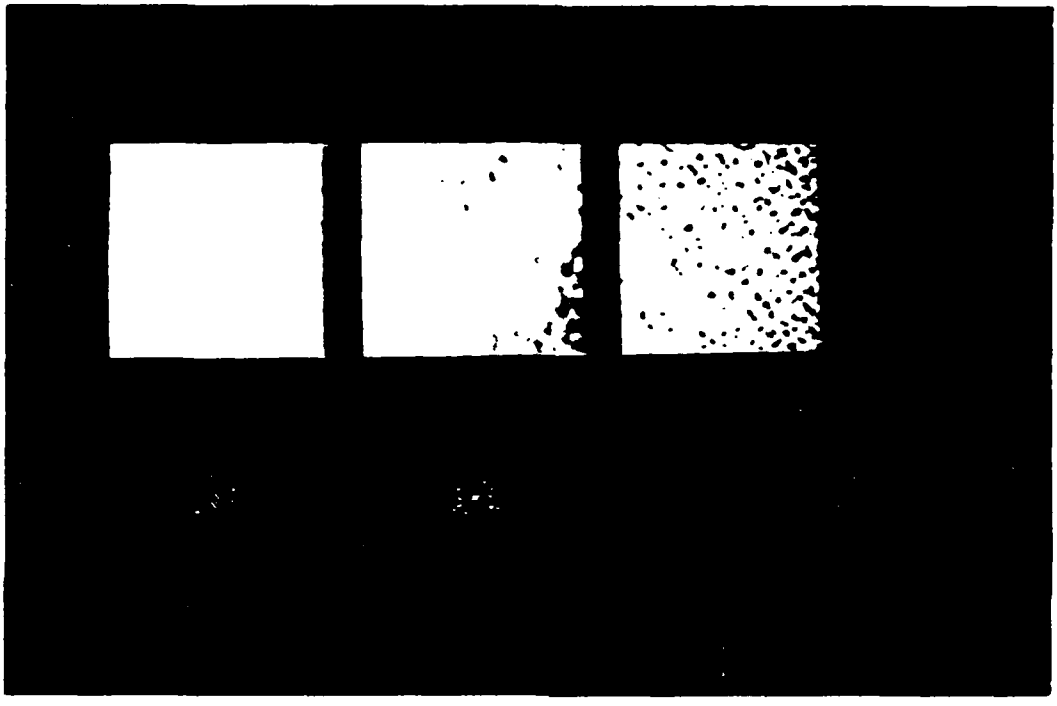


Fig. 13: Texture Recognition

lower left: The original image (BR2)
upper left: The power spectrum of the original image.
lower middle and right: Reconstructed images using different generation schemes (see text).
upper middle and right: Power spectrum of the reconstructed images.

Comment: The frequency component of the reconstructed image has been reduced slightly. The contrast in the generated images has decreased. By increasing the contrast, a better similarity may be achieved.

11 References

[1] Ballard, D.H. and Brown, C.M., *Computer Vision*, Prentice Hall, Englewood Cliffs NJ , 1982.

[2] Brodatz P., *Textures*, Dover Publications, New York, 1966.

[3] Eastman Kodak Quality Standards Organization Reference Standard for Coating Uniformity: Mottle, Eastman Kodak Company Publication, Rochester, NY.

[4] Haralick, R.M. "Statistical and structural approaches to texture," *Proceedings*, 4th International Joint Conference on Pattern Recognition, Kyoto, Japan 1978

[5] Hinkelman, Elizabeth A., "A stochastic Fourier domain mottle model," Internal report, Computer Science Dept., Univ. of Rochester, April 1986.

[6] Horn, B.K.P., *Robot Vision*, McGraw Hill Book Company, 1986.

[7] Knuth, D.E., *The Art of Computer Programming*, Vol 2, Addison-Wesley Publishing Company, 1981.

[8] Levine, M.D., *Vision in Man and Machine*, McGraw Hill Book Company, 1985.

[9] Rich E., *Artificial Intelligence*, McGraw Hill Book Company, 1986.

[10] Rosenfeld A. and Lipkin B.S., "Texture synthesis", in Lipkin, B.S. and Rosenfeld A. (eds), *Picture Processing & Psychopictorics*, Academic Press, NY, 1970.

On Relaxation Algorithms Based on Markov Random Fields*

Paul B. Chou
Rajeev Raman
Computer Science Department
The University of Rochester
Rochester, New York 14627

TR 212
July 10, 1987

ABSTRACT

Many computer vision problems can be formulated as computing the minimum energy states of thermal dynamic systems. However, due to the complexity of the energy functions, the solutions to the minimization problem are very difficult to acquire in practice. Stochastic and deterministic methods exist to approximate the solutions, but they fail to be both efficient and robust. In this paper, we describe a new deterministic method - the Highest Confidence First** algorithm - to approximate the minimum energy solution to the image labeling problem under the Maximum A Posteriori (MAP) criterion. This method uses Markov Random Fields to model spatial prior knowledge of images and likelihood probabilities to represent external observations regarding hypotheses of image entities. Following an order decided by a dynamic stability measure, the image entities make local estimates based on the combined knowledge of priors and observations. We show that, in practice, the solutions so constructed compare favorably to the ones produced by existing methods and that the computation is more predictable and less expensive.

*This work was supported in part by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, and the Air Force Office of Scientific Research, Bolling AFB, DC 20332 under Contract No. F30602-85-C-0008. This contract supports the Northeast Artificial Intelligence Consortium (NAIC). This work was also supported in part by U.S.Army Engineering Topographic Laboratories research contract no. DACA76-85-C-0001, in part by NSF Coordinated Experimental Research grant no. DCR-8320136, in part by ONR/DARPA research contract no. N00014-82-K-0193, and in part by a grant from the Eastman Kodak Company. We thank the Xerox Corporation University Grants Program for providing equipment used in the preparation of this paper.

**Also known as "He who hesitates is last".

1. Introduction

Probability theory has found many applications in representing the uncertainty of various kinds of knowledge and in reasoning about the world [Feldman and Yakimovsky 1974] [Duda, Hart, and Nilsson 1976] [Peleg 1980] [Pearl 1985] . It appeals to the AI community for many reasons, among which are that it provides a well-developed mathematical theory for using uncertainty measures in making decisions, and that it provides well-known ways of incorporating empirical data. However, there have been few successful attempts at utilizing this tool in practical machine vision systems. It is apparently not because this domain is any the less "uncertain" but because the complexity of the information in this domain hinders the advancement of probabilistic approaches.

It is our goal to demonstrate the practicability of applying the Bayesian-probability formalism to complex domains, such as the image labeling problem discussed in this paper. The *image labeling problem* is to assign labels to image entities such as regions, line segments, and pixels. The set of labels, usually reflecting the photometric and geometrical phenomena of the scene, is mutually exclusive and exhaustive at a particular level of abstraction. Denote the set of labels $\{ l_1, l_2, \ldots, l_Q \}$ as L, and the set of the image entities $\{ s_1, s_2, \ldots, s_N \}$ as S. Any mapping from S to L is a *feasible solution* to the labeling problem. To choose an "optimal" solution from the set of feasible solutions - Ω, image observations as well as prior knowledge about spatial relations between labels are used to evaluate the goodness of each solution. This work follows the probabilistic model for visual computation proposed in [Chou and Brown 1987b]. Spatial prior knowledge and local visual observations are separately represented in terms of probabilities. This decoupling provides a clean and uniform way of modeling information at different levels of abstraction, and therefore to modularize the design and implementation of probabilistic systems. Bayes' rule is used to combine priors and observations to form the *a posteriori* probabilities representing the updated knowledge. The labeling problem is then formulated as a minimization problem based on the Bayesian decision rationale. We shall show that by using a new algorithm proposed here to estimate the optimal solution to the minimization problem so formulated, it is possible to achieve excellent results with relatively little computation, given a set of reasonable assumptions.

The organization of this paper is as follows. We discuss how to encode the *a priori* knowledge of image events with the Markov random field (MRF) models in Section 2. The Bayesian decision rationale is discussed in Section 3. In Section 4, we review several stochastic relaxation methods that, in principle, could find the optimal solutions given enough computational resources. We describe a new deterministic estimation algorithm in Section 5 that our experiments indicate to be superior to existing methods. Experiments on edge detection - an instance of the labeling problem - with both synthetic and natural images are conducted with an MRF simulation/estimation package implemented by the authors. Results from various estimation schemes are compared in Section 6.

2. Markov Random Fields and Gibbs Distributions

Markov Random Fields have been used for image modeling in many applications for the past few years [Hassner and Slansky 1980] [Cross and Jain 1983] [Marroquin, Mitter, and Poggio 1985] [Geman and Geman 1984] [Derin and Cole 1986] [Chou and Brown 1987a]. In this section, we review the properties of MRF's and discuss how to

encode prior knowledge in this formalism. We refer the reader to [Kindermann and Snell 1980] for an extensive treatment of MRF's.

2.1. Noncausal Markovian Dependency

Let $X = \{X_s, s \in S\}$ denote a set of random variables indexed by S. Without loss of generality, assume all variables in X have a common state space L, so that $X_s \in L$. Let $\{X = \omega\}$ be the event $\{X_{s_1} = \omega_{s_1}, \ldots, X_{s_N} = \omega_{s_N}\}$, where $\omega = (\omega_{s_1}, \omega_{s_2}, \ldots, \omega_{s_N})$, $\omega_s \in L$, is a *configuration* of X. Since a configuration of X is also a feasible solution to the labeling problem, Ω also denotes the set of all possible configurations.

Let E be a set of unordered pairs (s_i, s_j)'s representing the "connections" between the elements in S. The semantics of the connections will become clear shortly. E defines a neighborhood system $\Gamma = \{N_s \mid s \in S\}$, where N_s is the neighborhood of s in the sense that

(1) $s \notin N_s$, and

(2) $r \in N_s$ if and only if $(s, r) \in E$.

X is a *Markov Random Field* with respect to Γ and P, where P is a probability function, if and only if

$$(\textit{positivity}) \quad P(X = \omega) > 0 \text{ for all } \omega \in \Omega \tag{2.1}$$

$$(\textit{Markovianity}) \quad P(X_s = \omega_s \mid X_r = \omega_r, r \in S, r \neq s) = P(X_s = \omega_s \mid X_r = \omega_r, r \in N_s) \tag{2.2}$$

The set of conditional probabilities on the left-hand side of (2.2) is called the *local characteristics* that characterizes the random field. It can be shown that the joint probability distribution $P(X = \omega)$ of any random field satisfying (2.1) is uniquely determined by these conditional probabilities [Besag 1974]. An intuitive interpretation of (2.2) is that the contextual information provided by $S - s$ to s is the same as the information provided by the neighbors of s. Thus the effects of members of the field upon each other is limited to local interaction as defined by the neighborhood. Notice that any random field satisfying (2.1) is an MRF if the neighborhoods are large enough to encompass all the dependencies.

2.2. Encoding Prior Knowledge and Gibbs Distributions

The utility of the MRF concept for image labeling problems is that the prior knowledge about spatial dependencies among the image entities can be adequately modeled with neighborhoods that are small enough for practical purposes. Very often, the image entities are regularly structured and prior distributions on the image are homogeneous and isotropic. In such cases, the number of parameters needed to specify the priors is just a fraction of Q^M, where M is the size of the neighborhoods. This is a significant saving over Q^N - the number of possible configurations, especially when M is small.

There are difficulties, as stated in [Geman and Geman 1984], associated with using the MRF formulation by itself:

(1) The joint distribution of the X_s is not apparent;

(2) It is extremely difficult to spot local characteristics, *i.e.*, to determine when a given set of functions are conditional probabilities for some distribution on Ω.

(1) is not a serious problem for some special classes of MRF models such as *Markov Mesh* (MM) processes [Kanal 1980], since their joint distributions can be represented in a recursive formulation due to the casual dependency assumed. For (2), parametric probability distributions such as Gaussian and binomial, have been been used in the literature [Cross and Jain 1983] [Cohen and Cooper 1987]. Using such distributions further simplifies the encoding of the local characteristics and has shown some impressive results on modeling and generating texture patterns. However, whether these kinds of simplifications preserve the power of MRF's for modeling spatial knowledge remains questionable.

Fortunately, these difficulties vanished when the following property of MRF's was realized.

Hammersley-Clifford Theorem: A random field X is an MRF with respect to a neighborhood system Γ if and only if there exists a function V such that

$$P(\omega) = \frac{e^{-\frac{1}{T}U(\omega)}}{Z} \quad \text{for all } \omega \in \Omega \tag{2.3}$$

where T and Z are constants and

$$U(\omega) = \sum_{c \in C} V_c(\omega). \tag{2.4}$$

C denotes the set of totally connected subgraphs (cliques) with respect to Γ. Z is a normalizing constant and is called the *partition function*.

The probability distribution defined by (2.3) and (2.4) is called a *Gibbs distribution* with respect to Γ. The class of Gibbs distributions has been extensively applied to model physical systems, such as ferromagnets, ideal gases, and binary alloys. When such systems are in a state of *thermal equilibrium* , the fluctuations of their configurations follow a Gibbs distribution. In statistical mechanics terminology, U is the *energy* function of a system. The V_c functions represent the *potentials* contributed to the total energy from the local interactions of the elements of clique c. T, the *temperature* of the system, controls the "flatness" of the distribution of the configurations.

Gibbs distributions, and therefore MRF's, possess a property that appears to be desirable for modeling - when constrained by a fixed expected value of some sufficient statistic of the random field, the *maximum entropy* distribution among the class of distributions compatible with the constraint is a Gibbs distribution.

The MRF-Gibbs equivalence not only relates the local conditional probabilities to the global joint probabilities, but also provides us a conceptually simpler way of specifying MRF's - specifying potentials. The importance of the joint probabilities will become evident in the next section. The local characteristics can be computed from the potential function through the following relation:

$$P(X_s = \omega_s \mid X_r = \omega_r, r \neq s) = \frac{e^{-\frac{1}{T}\sum_{c \in C_s} V_c(\omega)}}{\sum_{\omega'} e^{-\frac{1}{T}\sum_{c \in C_s} V_c(\omega')}} \tag{2.5}$$

where C_s is the set of cliques that contain s, and ω' is any configuration of the field that agrees with ω everywhere except possibly s.

There has been little work that applies statistical estimation methods to estimate parameters used for specifying MRF's. [Cross and Jain 1983] applies a coding scheme to estimate the parameters in their binomial distribution models using a maximum likelihood criterion. [Elliott and Derin 1984] uses a least-square-fit method to estimate potential functions in the Gibbs distributions of their texture models. These methods are good when many uncorrupted realizations are available. When such data are difficult to acquire, choosing the clique potentials on an *ad hoc* basis has been reported to produce promising results [Geman and Geman 1984] [Marroquin, Mitter, and Poggio 1985]. Our experiments (Section 6) have also shown good results. These results are not surprising since the notion of clique potentials provides a simple mapping from "qualitative" spatial knowledge to numeric values of the parameters specifying the MRF's.

3. Bayesian Decision Rationale and Optimality of Solutions

At various levels of a visual hierarchy, estimations (decisions) must be made based on the information available. The estimation procedures become complex when the information is uncertain, which is usually the case in visual processing. In this section, we examine the Bayesian decision rationale and the optimal solutions to the labeling problem with respect to this rationale.

3.1. *A Posteriori* Probabilities

Section 2 described how to encode prior spatial knowledge using the MRF formalism. Incorporating the image observations, Bayes' rule can then be used to derive the *a posteriori* probabilities on Ω from the *a priori* model of the image.

$$\text{(Bayes' Rule)} \quad P(\omega \mid O) = \frac{P(\omega)P(O \mid \omega)}{\sum_{\omega' \in \Omega} P(\omega')P(O \mid \omega')} \tag{3.1}$$

O denotes the image observations. The *likelihood* of an event $\{X = \omega\}$ given O, $P(O \mid \omega)$, is usually derived from the image degradation model involving imaging noise and blur [Geman and Geman 1984]. [Sher 1987] and [Bolles 1977] show methods to generate likelihood functions from either probabilistic models or statistical data.

For a shift-invariant point-spread function and white Gaussian noise, the *a posteriori* distribution associated with the *a priori* distribution defined by (2.3) and (2.4) is a Gibbs distribution with respect to a neighborhood system related to Γ and the support of the point-spread function [Geman and Geman 1984]. For simplicity, we assume the following conditional independence, that is generally true when the noise field is independently distributed.

$$P(O \mid \omega) = \prod_{s \in S} P(O_s \mid \omega_s) \tag{3.2}$$

O_s denotes a set of image observations over a spatial region dependent on s, typically including s and its spatially adjacent elements. This assumption appears to be very useful for fusing and modeling early visual modules [Chou and Brown 1987a] and for texture modeling [Derin and Cole 1986]. The *a posteriori* MRF thus has the same neighborhood

system Γ with the energy function

$$U_O(\omega) = \sum_{c \in C} V_c(\omega) - T \sum_{s \in S} \ln P(O_s | \omega_s) \tag{3.3}$$

3.2. Optimal Labelings

The goodness of a labeling $\hat{\omega}$, following the Bayesian formalism, is evaluated in terms of its *a posteriori* expected loss,

$$Loss(\hat{\omega} | O) = \sum_{\omega \in \Omega} loss(\hat{\omega}, \omega) P(\omega | O) \tag{3.4}$$

where $loss(\hat{\omega}, \omega)$ is a penalty associated with the estimate $\hat{\omega}$ while the "truth" is ω.

One question concerning the applicability of (3.4) is which loss function should be used for a given task. Except for few simple cases, the answer to this question usually relies on subjective judgements. One popular choice is assigning a constant penalty to incorrect estimates: $loss(\hat{\omega}, \omega)$ equals to a constant (positive) value whenever $\hat{\omega} \neq \omega$, and 0 otherwise. Using this loss function, the configuration minimizing (3.4) maximizes the *a posteriori* probability $P(\omega | O)$, and therefore minimizes the *a posteriori* energy (3.3) in the MRF formalism. This Maximum A Posteriori (MAP) criterion has been widely applied to the labeling problem [Feldman and Yakimovsky 1974] [Geman and Geman 1984] [Derin and Cole 1986] [Murray and Buxton 1987] [Cohen and Cooper 1987]. Marroquin et al [1985] suggest that the number of mislabeled image entities of an estimation is a better loss measure for the labeling problem. They derive the Maximizer of the a Posteriori Marginals (MPM) estimation - choosing the configuration $\hat{\omega} = (\hat{\omega}_{s_1}, \cdots, \hat{\omega}_{s_N})$ such that

$$\hat{\omega}_s = \max_{l \in L} P_s(l | O) \quad \forall s \in S, \tag{3.5}$$

where $P_s(l | O)$ denotes the *a posteriori* marginal probability of l on s. In their experiments the MPM estimator is shown to be superior to the MAP criterion when the signal to noise ratio is low.

Notice that the rationale of minimizing the loss function in (3.4) does not take the cost of computation into account, despite the fact that computational cost is usually a primary consideration in image understanding applications because of their immense configuration spaces. A sub-optimal estimator with an effective computation procedure would be much more useful than an optimal estimator that no one could ever compute. It is believed that the exact evaluation of MRF statistical moments, and therefore (3.4), is generally impossible since no analytic solutions exist [Hassner and Slansky 1980] [Geman and Geman 1984]. MAP and MPM can not be exactly determined for the same reason, except for some simple energy functions. In the rest of the paper, we discuss several numerical approaches for the approximate evaluations of the MAP and MPM estimations in the MRF formalism.

4. Stochastic Relaxation Methods

One method that has been successfully used to analyze the behavior of complex systems is generating sample configurations of a given system through stochastic simulations. Briefly, the Monte Carlo method of estimating the ensemble average of a variable $Y(\omega)$,

$$\langle Y \rangle = \int_{\Omega} Y(\omega) dP(\omega),$$

is averaging its values over a set of samples $\{ \omega_1, \cdots, \omega_R \}$ drawn from Ω. If the sampling of ω's follows the distribution P, then $\langle Y \rangle$ can be approximated by

$$\langle Y \rangle \approx \frac{1}{R} \sum_{r=1}^{R} Y(\omega_r).$$

We are interested in sampling procedures that generate configurations according to Gibbs distributions in the form of (2.3). With such procedures, the sample frequencies of the realizations of X_s can be used as approximations for the marginal probabilities, *i.e.*, MPM can be estimated; the configurations with higher probabilities are more likely to be sampled, and therefore MAP estimation becomes possible (see Section 4.2). Several procedures exist for this purpose. The basic idea of these procedures is to construct a regular Markov chain whose states correspond to the configurations of the system with the limiting distribution being the desired Gibbs distribution. That is, construct P_C - the transition matrix of the chain - in such a way that the following condition holds.

$$\pi P_C = \pi, \tag{4.1}$$

where π is the desired Gibbs measure. At equilibrium, the system's configurations are distributed according to π since π is the unique invariant measure of the constructed Markov chain [Kemeny and Snell 1960] .

Consider each state transition of the Markov chain involving only the change of the state of a single entity in the system. To fulfill the requirement of the chain being regular, the procedure must continue to "visit" every entity. Let $s(t)$ be the entity being visited at time t. The change of $X_{s(t)}$ would result a change of the system energy by the amount specified by the configurations of those cliques that contain $s(t)$ according to (2.4). Stochastic sampling procedures reminiscent of "relaxation" can be designed in the sense that the state transition of the entity being visited is stochastically decided by the states of the neighboring entities and itself. We will describe two of the *stochastic relaxation* procedures, namely the Metropolis algorithm [Metropolis et al. 1953] and the Gibbs sampler [Geman and Geman 1984], for their representativeness. Other variations basically follow the same principle and serve special purposes [Hassner and Slansky 1980] [Cross and Jain 1983] [Hinton and Sejnowski 1983] .

4.1. The Metropolis Algorithm and the Gibbs Sampler

Let $X(t)$ denotes the state of the system at time step t. The state transition from step t to $t+1$ of the Markov chain generated by the Metropolis sampling algorithm consists of two basic steps:

(1) Randomly select a new configuration ω' (randomly visit an entity s and choose a new state ω'_s), and compute the energy change $\Delta E = E(\omega') - E(X(t))$.

(2) If $\Delta E < 0$, set $X(t+1) = \omega'$. Otherwise, set $X(t+1)$ to ω' or $X(t)$ with probabilities $\frac{\pi(\omega')}{\pi(X(t))} = e^{-\Delta E/T}$ and $1 - e^{-\Delta E/T}$ respectively.

Allowing transitions with energy increases, a common characteristic of all stochastic relaxation procedures, prevents the sampling process from getting stuck at states of local

energy minimum - an undesirable property of every deterministic hill-climbing procedure. In contrast to the explicit use of the energy difference in the Metropolis algorithm, the Gibbs sampler uses the local characteristics to construct a Markov chain. A state transition of the Gibbs sampler also consists two steps:

(1) Visit an entity s.

(2) Randomly select the new state ω'_s for $X_s(t+1)$ following the distribution $\pi(X_s(t+1)=\omega_s \mid X_r(t), r \neq s)$. Having the form in (2.5), this distribution is generally easy to compute.

For binary systems, the Gibbs sampler is equivalent to the widely used "Heat Bath" algorithm - changing the state with probability $\frac{1}{1+e^{\Delta U/T}}$. Like other relaxation methods, the above procedures suggest the use of a parallel implementation since "updating" the X_s's requires propagating information only among neighboring computing units. Extra caution must be paid to the updating patterns of synchronous machines. For the Metropolis and Heat Bath algorithms, using any prescribed updating order may result in the Markov chain not converging to the desired Gibbs distribution π [Marroquin 1985]. Our experiments use the Gibbs sampler exclusively because it guarantees the coincidence of π with the invariant measure of the chain as long as neighboring entities are not updated simultaneously.

4.2. The Monte Carlo and Simulated Annealing Methods

The stochastic relaxation scheme can be used to approximate the *a posteriori* marginal probabilities for the MPM estimation by simulating the equilibrium behavior of the *a posteriori* MRF. Since the Markov chain constructed by either the Metropolis algorithm or the Gibbs sampler leads to the desired limiting distribution regardless of its initial state, the law of large numbers suggests the marginal probability $P_s(l \mid O)$ be approximated by the sample frequency of $X_s = l$ at equilibrium, that is,

$$P_s(l \mid O) \approx \frac{1}{n-k} \sum_{t=k}^{n} \delta(X_s(t)-l) \tag{4.2}$$

where $\delta(0) = 1$, and 0 elsewhere. k is the number of steps for the chain to reach equilibrium, and n is the total number of steps of the simulation. Practically, experimentation is needed to determine how large n and k should be to achieve a desirable approximation accuracy given an arbitrary MRF. Cross and Jain [1983] have observed that in less than 10 iterations (full sweeps over the image entities), their texture modeling system becomes "stable" when sampled by a variation of the Metropolis algorithm. In general, in the order of hundreds of iterations are needed for the MPM estimation.

The system temperature - T in (2.3) - also plays an important role in MRF simulations. With low temperatures, the Gibbs distribution strongly favors the low energy configurations, but the time required for the system to reach equilibrium may be long. The system may reach equilibrium faster at higher temperatures, but the configurations are more evenly sampled; *i.e.*, it may require more samples to make accurate MPM estimations. The idea of *simulated annealing* [Kirkpatrick, Gelatt, and Vecchi 1983], obviously inspired by physical annealing, is to reach the minimum energy states of a system by starting the system at a high temperature and gradually reducing it. In doing so the system tends to respond to large energy differences at the beginning, and is likely to find a good minimum

energy state independent of its starting state. As the temperature decreases, the system tends to respond to small energy differences, and ideally settles at the lowest energy states ever encountered. The decreasing sequence of temperatures, called the annealing schedule, decides the effectiveness of this process. If the time spent at each temperature is not enough, the system may not converge to the global minimum states. On the other hand, it is often computationally prohibitive to use a slowly decreasing schedule. Geman and Geman [84] have derived an upper bound for the annealing schedules so that the schedules slower than this bound are guaranteed to converge to the global minimum energy states. However, this bound is very difficult to decide in practice since it relates to the range of energy values of the system.

Simulated annealing has been applied in many computer vision tasks that involve optimization over exponential spaces, including the MAP estimation [Geman and Geman 1984] and the stereo matching problem [Barnard 1987]. One major concern of using the stochastic relaxation scheme is its efficiency: at what cost can this scheme deliver satisfactory results? Not surprisingly, the cost is intolerable for many applications. In the next section, we describe a new deterministic method to approximate MAP. This method, following a search path suggested by the visual observations to find a minimum energy state, appears to give results favorably comparable in practice to the existing relaxation methods while being computationally less expensive.

5. Deterministic Relaxation Methods

Exact calculation of the MAP estimate is computationally prohibitive. For vision systems that require predictable results in reasonable time periods, using suboptimal estimation criteria and/or heuristics in searching for solutions seems to be a reasonable alternative to the stochastic relaxation scheme. In [Derin and Cole 1986], MAP estimations are performed on narrow strips of the image. The strips are limited to at most four rows wide so that MAP can be exact computed for each strip by a dynamic programming algorithm at feasible cost. For each estimation, only the estimate of the first row of a strip is kept. It serves as the boundary condition for the next strip consisting of the rest of the rows and a new one. Though limiting the extent of the (column-wise) interactions, the texture segmentation results appear to be impressive. Before we describe the proposed heuristic-based algorithm, we examine an iterative relaxation method for estimating MAP.

5.1. Iterative Energy Minimization

A simple version of deterministic iterative relaxation methods for energy minimization is the Metropolis algorithm without randomness: Start with an initial configuration. At each iteration through the image entities, the state of each entity is either changed to the state that yields maximal decrease of the energy, or is left unchanged if no energy reduction is possible. The process stops when no more changes can be made. This algorithm is guaranteed to find a local minimum of the energy function since each iteration strictly decreases the energy value and there are only a finite number of different values of the energy function. For parallel implementation, convergence is assured if the neighboring entities are not updated simultaneously.

Unavoidably, the local minimum obtained by the above algorithm may be far from optimal. Two enhancements are apparently helpful:

(1) Start with a better initialization of the MRF. The best one can hope is that the energy value of the initial configuration falls into the valley of the global minimum. One possibility is to use the *maximum likelihood estimates* (MLE) - $X_s(0)=\omega_s$ if $\max_{l\in L} P(O_s|l)=P(O_s|\omega_s)$.

(2) Escape from shallow valleys. By changing the states of more than one entity at once, the new configuration may lead to a better local minimum. In a procedure described in [Cohen and Cooper 1987], the entities with small preferences of the current states over the others are assigned new states when a local minimum is reached. The relaxation restarts with the new configuration as the initialization. At each convergence, the magnitude of the local minimum is estimated, The procedure halting when no significant change of the magnitudes is observed. The hope is that the deepest valley will be found in this process.

Unfortunately, these two modifications are not adequate. The local MLE's are good only when the noise process is correctly modeled in computing the likelihoods and there are significant differences among the likelihoods of the hypotheses. Frequently these conditions cannot be met. Cohen and Cooper's procedure, obviously a compromise between stochastic and deterministic relaxation methods, suggests a tradeoff between speed and performance.

The algorithm of this paper blends the initialization into the estimation process. Instead of stepping through the configuration space Ω, this algorithm constructs a configuration with a local minimal energy measure. Observable evidence and spatial prior knowledge are combined in the process of the construction, resulting in better results and efficiency. The details of this algorithm are described next.

5.2. The Highest Confidence First Algorithm

To see how this algorithm works, some terminology needs to be introduced. Let $\bar{L} = L\cup\{l_0\}$ denote the *augmented label set*, where $L = \{l_1, \cdots, l_Q\}$ is the set of labels, and l_0 is the null label corresponding to the "uncommitted" state in the construction. Let $\bar{\Omega} = \{\omega=(\omega_1, \cdots, \omega_N)\,|\,\omega_s\in\bar{L}, \forall s\in S\}$ denote the *augmented configuration space*. Define the *augmented a posteriori local energy* of $l\in L$ with respect to $s\in S$ and a configuration $\omega\in\bar{\Omega}$ as

$$E_s(l)=\sum_{c:\,s\in c} V'_c(\omega') - T\,\ln P(O_s|l), \tag{5.1}$$

where $\omega'\in\bar{\Omega}$ is the configuration that agrees with ω everywhere except $\omega'_s = l$, and V'_c is 0 if $\omega_r = l_0$ for any r in c, otherwise it is equal to V_c - the potential function.

The basic idea of this algorithm is to construct a sequence of configurations $\omega^0, \omega^1, \cdots$ with the starting configuration $\omega^0 = (l_0, \cdots, l_0)$, and a terminal configuration $\omega^f\in\Omega$, where $U_O(\omega^f)$ is a local minimum with respect to Ω. We say an entity s has *made* its *current decision* l, $l\in L$, if the just-constructed (current) configuration ω in the sequence has the component $\omega_s = l$, and it has not made a decision if $\omega_s = l_0$. Once an entity makes a decision, it can *change* this decision to other labels of L but not to l_0. To ensure the quality of the resulting estimate - ω^f, at each step of the construction we permit only the least "stable" entity to change/make its decision. We define the *stability* of s with respect to the current configuration ω, $\omega_s = l$, as

$$G_s(\omega) = \min_{k \in L, k \neq l} \Delta E_s(k,l) \quad \text{if } l \in L \tag{5.2a}$$

$$G_s(\omega) = -\min_{k \in L, k \neq j} \Delta E_s(k,j) \quad \text{if } l = l_0 \text{ and } j \in L \text{ s.t. } E_s(j) = \min_{k \in L} E_s(k), \tag{5.2b}$$

where $\Delta E_s(j,k) = E_s(j) - E_s(k)$ with respect to ω.

The stability defined above is a combined measure of the observable evidence and the *a priori* knowledge about the preferences of the current state over the other alternatives. A negative value of G, that is always true for (5.2b), indicates a more stable configuration will result from an alternative decision. Since an entity has no effect on its neighbors unless it has made a decision, the entities with large likelihood ratios of one label over the others - strong external evidence in favor of a label - will be visited early in the construction sequence. The entities with little idea from the observations will collect information from the neighbors' decisions to make their decisions; an early decision will be altered if the neighbors' later decisions are strongly against it. In this way, every step of this construction makes a maximal progress based on the current knowledge about the field - the G's. This *Highest Confidence First* algorithm is expected to find the estimate that is "consistent" with the observations and the *a priori* knowledge.

The Highest Confidence First algorithm can be implemented serially with a heap (priority queue) maintaining the visiting order of the construction according to the values of G's in such a way that the *top* of the heap is the entity with the smallest G value. Updating the *top*'s decision will cause the changes of its neighbors' G-values, and therefore the structure of the heap. The following is the pseudo code for the Highest Confidence First algorithm:

```
ω = (l_0, ···, l_0);
top = Create_Heap(ω);
while (G_top < 0) {
    s = top;
    Change_State(ω_s);
    Update_G(G_s );
    Adjust_Heap(s);
    foreach (r ∈ N_s) {
        Update_G(G_r);
        Adjust_Heap(r);
    }
}
return(ω);
```

Change_State(ω_s) changes the current state ω_s of s to the state l such that $\Delta E_s(l, \omega_s) = \min_{k \in L, k \neq \omega_s} \Delta E_s(k, \omega_s)$ if $\omega_s \in L$, or $E_s(l) = \min_{k \in L} E_s(k)$ if $\omega_s = l_0$. Upon this change taking place, the stability of s changes to positive. Update_G is called for every entity that is affected by this change, namely the neighbors of s according to (5.1), to update their stability measures with respect to the new configuration. Adjust_Heap(r) maintains the heap property by moving r up or down according to its updated G-value.

Several desirable properties of this procedure can easily be verified:

(1) Termination: This procedure always returns in finite time. To see this property, let us consider the two types of Change_State - making and changing a decision - separately. The procedure can make at most N decisions, one for each entity, since nullifying decisions is impossible. Let $D = (S_D, S - S_D)$ be a *partition* of S such that S_D is the set of entities that have made decisions. Let $\overline{\Omega_D} = \{\omega \in \bar{\Omega} \mid \omega_s \in L \ \ \forall s \in S_D, \text{ and } \omega_s = l_0 \ \ \forall s \in S - S_D\}$. Since, by (5.1) and (5.2a), changing the decision of $s \in S_D$ strictly decreases the function $U_D : \overline{\Omega_D} \rightarrow R$,

$$U_D(\omega) = \sum_c V'_c(\omega) - T \sum_{s \in S_D} \ln P(O_s \mid \omega_s),$$

the procedure can make only a finite number of changes with respect to a fixed partition D. There are only a finite number of partitions, therefore the total number of decision changes is finite.

(2) Feasibility: The returned configuration is in Ω - the space of feasible solutions. For if otherwise, there exists an s such that $\omega_s = l_0$. From (5.2b), $G_s < 0$. This violates the heap invariant property since it requires $G_{top} \geq 0$ to exit the while loop.

(3) Optimality: The returned configuration has the locally minimal energy measure with respect to Ω. That is, changing the decision of any single entity can not decrease the *a posteriori* energy measure U_D. As above, this property can easily be derived from (5.2a) and the heap properties.

This implementation takes $O(N)$ comparisons to create the heap and $O(\log(N))$ to maintain the heap invariance for every visit to an entity, provided the neighborhood size is small relative to N. The overheads of heap maintenance are well repaid since the procedure makes progress for every visit, in contrast to the iterative relaxation procedure (Section 5.1) that may make only few changes per iteration (N visits). Our experiments show that on the average, less than one percent of the entities are visited more than once using the proposed algorithm while the deterministic relaxation procedure takes around 10 iterations to reach a local minimum. This advantage becomes more evident as the number of entities gets larger. Experimental results are strongly in favor of the proposed algorithm for both efficiency and correctness. They are discussed in Section 6.

5.3. Possible Extensions

Since the order of the deterministic decisions of the entities in a cooperative network is crucial to the final mutual agreement, the proposed algorithm assumes that good results can be obtained by delaying the decisions of those entities who have little idea about what to do until they get enough help from their neighbors. This heuristic can be used along with other computational methods to achieve, perhaps, better results.

Let us look more closely at the process of achieving a consensus using this heuristic. At each stage, S_D consists of a set of isolated clusters. A cluster is a set of spatially connected (with respect to Γ) entities. We say two clusters are isolated from each other if none of the entities of a cluster is a neighbor of any entity of the other cluster. Each cluster corresponds to an MRF with free boundaries in our formalism. When an entity makes a decision, a cluster is created or expanded, or clusters are merged. When an entity changes a decision, the energy of the corresponding MRF is reduced. Eventually, all the clusters are merged and the final agreement corresponds to a local minimum configuration of the

corresponding MRF.

The notion of growing clusters suggests a natural partition of the image. At any instance, the entities belonging to the same cluster are tightly related, but they are independent of the members of other clusters. The addition of a new member to a cluster may change the decisions of the old members, but the changes are expected to be small due to the way the clusters are constructed. Therefore, it makes sense to compute the MAP estimates exactly for small clusters early in the process. We believe that by doing so the results would be better than the results using the horizontal strip partition as in [Derin and Cole 1986].

The process of growing clusters is similar to annealing in the sense that it responds to large energy differences earlier than small ones. Nondeterminism can be introduced to those entities that stay "unstable" - the entities on or exterior to the border of the clusters - late in the process, since more spatial information is required for them to reach a globally satisfactory agreement.

The Highest Confidence First algorithm can be implemented with a set of cooperative computing units. Consider a *winner-take-all* network where each unit corresponds to an entity of the image [Feldman and Ballard 1981]. Only the units with the smallest stability measures can "fire" at one instant; each unit maintains the knowledge about the neighboring units so that its stability measure can be updated immediately should any neighbor change its state. The parallelism gained, however, is limited due to the sequential firing order.

6. Experiments and Results

We have chosen to tackle the well studied problem of edge detection using MRFs as the underlying formalism. The labeling problem in this context is to assign to each edge element a label from the set {EDGE, NON-EDGE}. Each of these edge elements is modeled as an MRF entity. The MRF entities are considered to be situated on the boundary between two pixels (see Fig. 1). The MRF model used is similar to the "Line Process" MRF used both by Geman *et al* [1984] and Marroquin *et al* [1985]. Hence the MRF is binary, with $2(N^2-N)$ entities where the image is a $N \times N$ rectangular pixel array.

6.1. Construction of Potential Functions to Encode Prior Knowledge

The spatial relationships between entities we wish to enforce include:

(1) To encourage the growth of continuous line segments,

(2) To discourage abrupt breaks in line segments,

(3) To discourage close parallel lines (competitions) and

(4) To discourage sharp turns in line segments.

A second order neighborhood turns out to be sufficient to enforce all the relationships we want. In this neighborhood system, each MRF element is adjacent to eight others (see Fig.s 1 and 2).

The second order neighborhood has cliques of sizes 1 through 4 (see Fig. 3). The potential values we assign to various configurations of these cliques are shown in Fig. 4. These values form the specification of the potential functions. Therefore potential functions can be

seen to be specified by about 10 parameters, which are currently assigned in an *ad hoc* manner. The rules of thumb that are used to assign values to these parameters are:

Determine Structure Enforcers For each clique, attempt to determine what kind of structural relation it is uniquely capable of enforcing.

Encode Prior Structural Knowledge By assigning "high" potential values to undesirable configurations of the cliques and "low" values to desirable ones, we attempt to ensure that the final estimate will contain as few of the undesirable ones as possible.

Encode Statistical Prior Knowledge We use the clique consisting of the singleton node to bring the first order statistics (*e.g.* the density of EDGEs) of the MRF into line with what we already know. The potential of the clique when the MRF entity is an EDGE is set to our estimate of the log of the (local) odds of an entity being an EDGE over a NON-EDGE, and is set to 0 when it is a NON-EDGE.

A point to be noted is that some of these parameter values are interdependent. For example, increasing the energy for "break" (Fig. 4b) and "continuation" (Fig. 4c) configurations simultaneously would be of little use, as the increases would tend to cancel each other out.

The sensitivity of the results obtained to changes in the parameters specifying the potential functions depends upon the parameter in question. Our experience is that changing the potential function associated with the 1-clique had the greatest effect on the final result, followed by the 2-clique and 4-clique potential functions, in that order. This could be because the singleton clique controls first order statistics and the larger cliques higher order statistics, which are known to be less important in distinguishing images [Julesz 1981].

6.2. Likelihood Generation

We adopt a step-edge with white Gaussian noise model to compute the local likelihoods of an entity s being EDGE or NON-EDGE - $P(O_s|\omega_s=\text{EDGE})$ and $P(O_s|\omega_s=\text{NON}-\text{EDGE})$. The observation - O_s - is a 1×4 or 4×1 window of brightness observations surrounding s. This window of intensity values is assumed to be a realization of one of the possible e.nts depicted in Figure 5, corrupted by independent Gaussian noise. The reader is referred to [Sher 1987] for details of probabilistic edge detection.

From (3.2), observe that scaling $P(O_s|l)$ for every $l\in L$ by a constant factor for fixed s does not change the *a posteriori* distribution. This fact allows us to use the likelihood ratios - $\frac{P(O_s|\omega_s=\text{EDGE})}{P(O_s|\omega_s=\text{NON}-\text{EDGE})}$ - as the only input data, thus simplifying the computation of the stability measures (5.2). Thresholding the likelihood ratios by the prior (local) odds of an entity being an EDGE results in the thresholded likelihood ratio (TLR) configuration that can be considered as an MAP estimate obtained without using contextual information. In our experiments, we use TLR as the initial configuration whenever possible.

6.3. A General Purpose MRF Simulator

Our experiments use an interactive general-purpose MRF simulator package with extensive graphics and menu-driven control (Fig. 6). This package takes the description of the MRF and the likelihood ratios as input and simulates the state transitions of the

entities comprising the the MRF. The user can specify the estimation algorithm to be used and also the initialization of the MRF - each entity can be initially set to either a NON-EDGE or to its TLR state. The input MRF is constrained to be a homogeneous one, so as to make the time and space needed to run simulations reasonable.

The user provides the description of the MRF to the simulator in a file. This file contains a specification of the nodes comprising each clique, as well as the potential function associated with it. The user specifies all cliques that a node could belong to in the most general case, including all instances of a particular clique type that contain the node. (*i.e.*, even if all the cliques containing a node are instances of the same clique type, the user specifies each instance separately). The nodes forming a clique are specified by their coordinates relative to the node of interest, which is defined to be at relative coordinates (0, 0). Boundary conditions, as in the case of nodes near the border of the MRF, are taken care of by the simulator. The potential function is specified as a function that takes as input a configuration vector (a vector of states of nodes of the MRF) and returns a potential value. The potential function is associated with the clique description, and the ordering of the node states in the configuration vector passed it is the same as the order the nodes are specified in the description of the clique itself.

The simulator performs certain preprocessing actions on the description of the MRF provided by the user, to promote run-time efficiency. The first is to store each potential function as a table indexed into by a configuration vector. This is done so as to avoid run-time calling of the user's potential function code , which can be quite complex, replacing it instead with a simple table lookup. The other is "clique containment", which is based on the observation that if one clique completely contains the other, then a configuration vector of the nodes in the larger clique contains implicitly the configuration vector for the smaller clique. This suggests that by judiciously "adding" together the potential functions for the clique in the preprocessing stage, we can avoid run-time evaluation of the potential function for the smaller clique. This simplifies the state transition energy evaluation by reducing the number of terms to be summed up. If floating-point arithmetic is costly, this can save considerable computational effort. The preprocessing needs to be done just once, and can be performed off-line.

6.4. Experimental Results

The simulator described above has been used for a series of experiments aimed at comparing the performances of various relaxation algorithms with respect to the goodness of final estimations and rate of convergence. We focus upon algorithms using the MAP criterion, including Highest Confidence First (HCF), Deterministic Iterative Relaxation (DIR) and stochastic MAP (simulated annealing with Gibbs Sampler [Geman and Geman 1984]). The results obtained by using stochastic MPM (Monte Carlo approximation to the MPM estimate [Marroquin, Mitter, and Poggio 1985]) are also presented for the sake of completeness of comparisons, as are those obtained by applying 3×3 Kirsch operators with non-maximum suppression. The annealing schedule for the stochastic MAP follows the one suggested in [Geman and Geman 1984], *i.e.* $T_k = \frac{c}{\log(1+k)}$ where T_k is the temperature for the k^{th} iteration, with $c = 4.0$. The stochastic MAP was run for 1000 iterations and the stochastic MPM for 500 (300 to reach equilibrium, 200 to collect statistics).

6.4.1. Comparison of Estimates

Here we show the results of three sets of experiments (Fig.s 7 through 9). The figures for each set contain the original image, the result from the Kirsch operators, the TLR configuration and the results obtained by using stochastic MAP, stochastic MPM, DIR (scan-line visiting order), DIR (random visiting order) and HCF algorithms. Except in the case of the HCF algorithm, where the MRF is initialized to all null (uncommitted) states, the MRF is initialized to the TLR configuration. The MRF specification is the same throughout.

Fig. 7a shows a synthetic 50 pixel square "checkerboard" pattern. Each of patches is 10 pixels across, with an intensity chosen randomly from between 0 and 255. The image has been degraded by independently adding to each pixel Gaussian noise with a mean of 0 and a standard deviation of 16. The HCF and stochastic MPM (Figs. 7g and 7d) are the same, and have completed most of the desired edges. The DIRs (Figs. 7d and 7e) have incomplete edges and the stochastic MAP has some undesired edges and incomplete desired edges (Fig. 7c). The Kirsch operator result is not shown as the edges in this image are always located exactly in between pixels, while the Kirsch operator assumes edges to be at pixel locations, and so a comparison would be unfair to the Kirsch operators.

Fig. 8a shows a 50 pixel square natural image of a wooden block with the letter "P" on it. The MAP estimate has several undesirable lines (Fig. 8d). The MPM estimate performs poorly on the right edge of the block and the inner ring of the "P". The DIR scheme (serial scan) (Fig. 8f) performs better than the random scan version (Fig. 8g), but is less than satisfactory on the leg of the "P" and the right edge of the block. The HCF estimate (Fig. 8h) does not suffer from the above flaws, producing clean, connected edges.

Fig. 9a shows a 100×124 natural image of 4 plastic blocks with the letters "U", "R", "C" and "S" on them. Again, the HCF algorithm produces superior results (Fig. 9g). It has the clearest letter outlines and also is alone in detecting the entire bottom edge of the "R" block. The MAP estimate partially detects the bottom edge of the "R" block, but generates redundant lines (Fig. 9c). The MPM estimate has clear letter outlines but does poorly on the outlines of the left blocks (Fig. 9d). The DIR scheme (scan-line) does well on the letter outlines but poorly on the block outlines while the random scan version does poorly on both (Figs. 9e and 9f).

To test the *robustness* of the algorithms, we conduct further experiments using a likelihood generator with a less complete edge model. Since offset edges (Fig. 5c) are not considered here, multiple responses become significant as can be seen from the TLR configuration shown in Fig. 10a. This change strongly affects the estimates produced by all the algorithms except the HCF, as can be seen from comparing corresponding pictures in Fig. 9 and Fig. 10.

6.4.2. Rates of Convergence

We restrict ourselves to comparisons between deterministic schemes, as stochastic schemes do not have any convergence criterion *per se* - the point of convergence is dependent upon our judgement as to when equilibrium has been reached, and as to when we have gathered enough statistics to estimate the joint (or marginal) probabilities accurately (typically several hundred iterations are needed). The deterministic algorithms (HCF and

DIR (scan-line)) have been timed on images of various sizes using a Sun 3/260 with floating point acceleration. The results are shown in Table 1.

6.5. Analysis of Experimental Results

Goodness of Estimates

(1) The HCF algorithm repeatedly outperforms all other algorithms, giving superior results both with synthetic and real image data. The common characteristics of the results we have obtained from using this algorithm are that they all fit well in our model of the world, which consists of smoothly continuous boundaries, and that they are consistent with the observations.

(2) The HCF algorithm also appears to be robust, in that it produces an estimate consistent with the observations even when the MRF model used is inadequate, as in the experiment using the less sophisticated edge detector. Since our MRF model does not take into account multiple responses, the MAP criterion may not lead to the "best" results. In this case, the local minimum found by the HCF algorithm is actually better than the global one as it is based on the strength of external evidence.

(3) The DIR algorithm performs inconsistently and its results depend to a large extent upon the initialization of the MRF and the visiting order. It is also not clear which, if any, of the visiting orders studied is better than the other.

(4) The stochastic MAP algorithm with simulated annealing gets stuck in undesirable local minima, suggesting that our annealing schedule might have lowered the temperature too fast. However, an appropriate annealing schedule seems hard to obtain *a priori*.

Convergence Times

(1) The HCF algorithm makes a perhaps surprisingly small number of visits before converging. Clearly, due to the initialization, it must visit every node at least once. What is surprising is that it visits each node on the *average* less than 1.01 times before converging. What this implies is that the first decision made by a node is nearly always the best one.

(2) The convergence times of the DIR algorithms are unpredictable - they vary with visiting order, MRF initialization and even upon the particular image given as input. The HCF algorithm, in contrast, takes almost the same time on different images of the same size. The time taken by the HCF algorithm includes the time taken to set up the heap initially. This may, in some circumstances, be a little unfair. For instance, if one has to process data online from various information sources [Chou and Brown 1987a] [Poggio 1985], the heap setting up cost can be treated as a preprocessing cost rather than a run-time one.

(3) In theory, the time taken by the HCF algorithm should be given by $c_1 N + c_2 V \log_2 N$, where c_1 and c_2 are positive constants, N the number of entities to be labeled and V the number of visits. V here is at least N and we conjecture that on the average it is cN for some small $(1<c<2)$ constant c. Since the latter term should dominate, one would expect to see a nonlinear curve in a plot of run time *vs.* number of entities. However, the curve is very nearly a straight line. which indicates either that the

constant c_2 is very small, or that the changed stability values do not propagate very far up the heap on the average. The former does not appear to be true, as our experiences suggest that the initial heap construction takes far less time than the rest of the algorithm.

7. Conclusions and Future Research

We have described a new approach for solving the labeling problem. The Highest Confidence First algorithm, aimed at approximating the MAP estimate with *a priori* knowledge modeled by MRF's and external observations represented as likelihoods, leads to outstanding results in our experiments with both synthetic and natural images. Not only is this algorithm much faster than stochastic estimation procedures, it also converges predictably. In addition, the algorithm is robust - in the case that the prior model proves inadequate, it produces an estimate that is highly consistent with the observations.

We are incorporating the Highest Confidence First algorithm in a multi-modal segmenter described in [Chou and Brown 1987a] and believe it to be well suited to a scenario where the result is to be computed incrementally from sparse and dynamically-arriving data, possibly from multiple sources.

We are studying methods for systematically specifying the clique potential functions of MRF's from given realizations. We are also analyzing the rapid convergence of the HCF algorithm observed in our experiments from a theoretical viewpoint.

The concept of a confidence-based heuristic is likely to be useful whenever there is a set of cooperating processes attempting to reach a consensus. The idea that the processes with a greater degree of certainity about their decision, get to make it first, is intuitively appealing. We are investigating applications of this idea to other fields.

Acknowledgements

We thank Chris Brown for helpful discussions on this subject. Without his active encouragement and support this work would have been impossible.

We are indebted to Dave Sher for suggesting the low-level edge models used in this work, and for the use of edge-detection software developed by him for our experiments.

The first author would also like to express his gratitude to the members of his thesis committee (Dana Ballard, Chris Brown, Jerry Feldman and Henry Kyburg) for their valuable suggestions.

References

Barnard, S., "Stereo Matching By Hierarchical, Microcanonical Annealing", *Proceedings: Image Understanding Workshop 2* (Feb. 1987), 792-797.

Besag, J., "Spatial interaction and the statistical analysis of lattice systems (with discussion)", *Journal of Royal Statistics Society, series B 36* (1974), 192-326.

Bolles, R. C., "Verification Vision for Programmable Assembly", *Proceedings: IJCAI-77*, Aug. 1977, 569-575.

Chou, P. B. and C. M. Brown, "Probabilistic Information Fusion for Multi-Modal Image Segmentation", *Proceedings: IJCAI-87*, Milan, Italy, Aug. 1987.

Chou, P. B. and C. M. Brown, "Multi-Modal Segmentation Using Markov Random Fields", *Proceedings: Image Understanding Workshop 2* (Feb. 1987), 663-670.

Cohen, F. S. and D. B. Cooper, "Simple Parallel Hierarchical and Relaxation Algorithms for Segmenting Noncausal Markovian Random Fields", *IEEE Tansactions on Pattern Analysis and Machine Intelligence PAMI-9*, 2 (Mar. 1987), 195-219.

Cross, G. R. and A. K. Jain, "Markov Random Field Texture Models", *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5*, 1 (Jan. 1983), 25-39.

Derin, H. and W. S. Cole, "Segmentation of Textured Images Using Gibbs Random Fields", *Computer Vision, Graphics, and Image Processing 35* (1986), 72-98.

Duda, R. O., P. E. Hart, and N. J. Nilsson, "Subjective Bayesian Methods for Rule-Based Inference Systems", SRI Technical Note 124, SRI International, 1976.

Elliott, H. and H. Derin, "Modeling and Segmentation of Noisy and Textured Images Using Gibbs Random Fields", #ECE-UMASS-SE84-1, University of Massachusetts, Sept. 1984.

Feldman, J. A. and Y. Yakimovsky, "Decision Theory and Artificial Intelligence: I. A Semantics-Based Region Analyzer", *Artificial Intelligence 5* (1974), 349-371.

Feldman, J. A. and D. H. Ballard, "Computing with connections", TR 72, Computer Science Department, Univ. of Rochester, 1981.

Geman, S. and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-6*, 6 (Nov. 1984), 721-741.

Hassner, M. and J. Slansky, "The Use of Markov Random Fields as Models of Texture", in *Image Modeling*, Rosenfeld, A. (editor), Academic Press, Inc., 1980, 185-198.

Hinton, G. E. and T. J. Sejnowski, "Optimal Perceptual Inference", *Proceedings: IEEE Conf. CVPR*, 1983, 448-453.

Julesz, B., "Textons, the elements of texture perception, and their interactions", *Natural 290 12* (March, 1981), 91 - 97.

Kanal, L. N., "Markov Mesh Models", in *Image Modeling*, Rosenfeld, A. (editor), Academic Press, Inc., 1980, 239-243.

Kemeny, J. G. and J. L. Snell, *Finite Markov Chains*, Van Nostrand, New York, 1960.

Kindermann, R. and J. L. Snell, "Markov Random Fields and their Applications", in *Contemporary Mathematics*, vol. 1 , American Mathematical Society, 1980.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing", *Science 220* (1983), 671-680.

Marroquin, J., S. Mitter, and T. Poggio, "Probabilistic Solution of Ill-Posed Problems in Computational Vision", *Proceedings: Image Understanding Workshop*, Dec. 1985, 293-309.

Marroquin, J. L., "Probabilistic Solution of Inverse Problems", AI-TR 860, MIT Artificial Intelligence Laboratory, Sep. 1985.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equations of State Calculations by Fast Computing Machines", *Journal of Chemical Physics 21* (1953), 1087-1091.

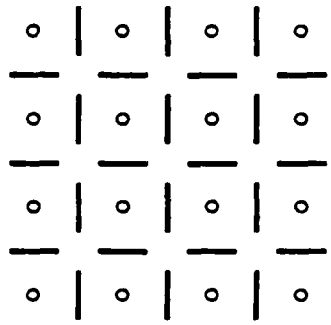
Murray, D. W. and B. F. Buxton, "Scene Segmentation from Visual Motion Using Global Optimization", *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-9*, 2 (Mar. 1987), 220-228.

Pearl, J., "Fusion, Propagation, and Structuring in Bayesian Networks", Computer Science Dpt.-850022 R-42, Apr. 1985. Computer Science Dept., UCLA.

Peleg, S., "A new probabilistic relaxation scheme", *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-2* (1980), 362.

Poggio, T., "Integrating vision modules with coupled MRFs", Working Paper No. 285, MIT A.I. Lab., Dec. 1985.

Sher, D. B., "Advanced Likelihood Generators for Boundary Detection", TR197, Univ. of Rochester, Computer Science Dpt., Jan. 1987.



: Line (MRF) Element

: Pixel (Image)

Figure 1

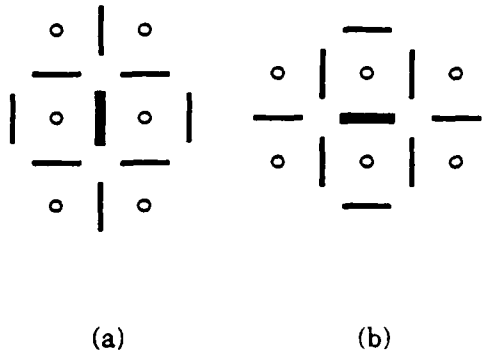


Figure 2

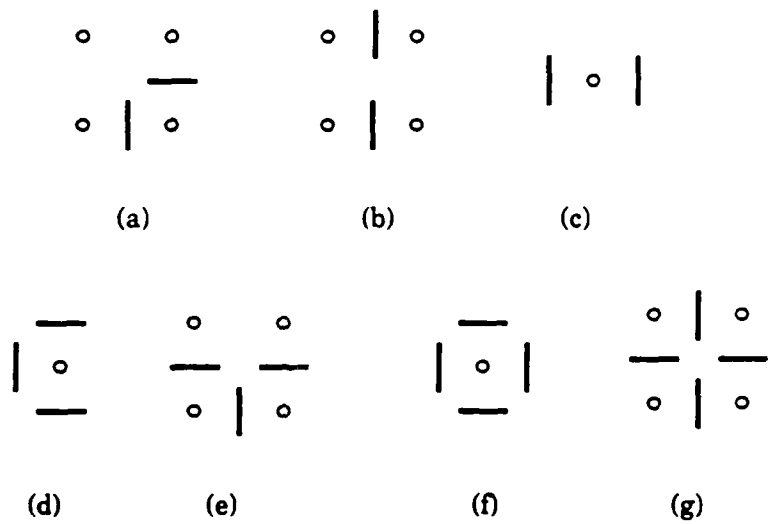


Figure 3

Figure 1: Relationship between MRF entities and pixels. **Figure 2**: The second-order neighborhood system. **Figure 3**: Cliques in neighborhood system, of size greater than one. (a)-(c): size 2; (d)-(e): size 3; (f)-(g): size 4

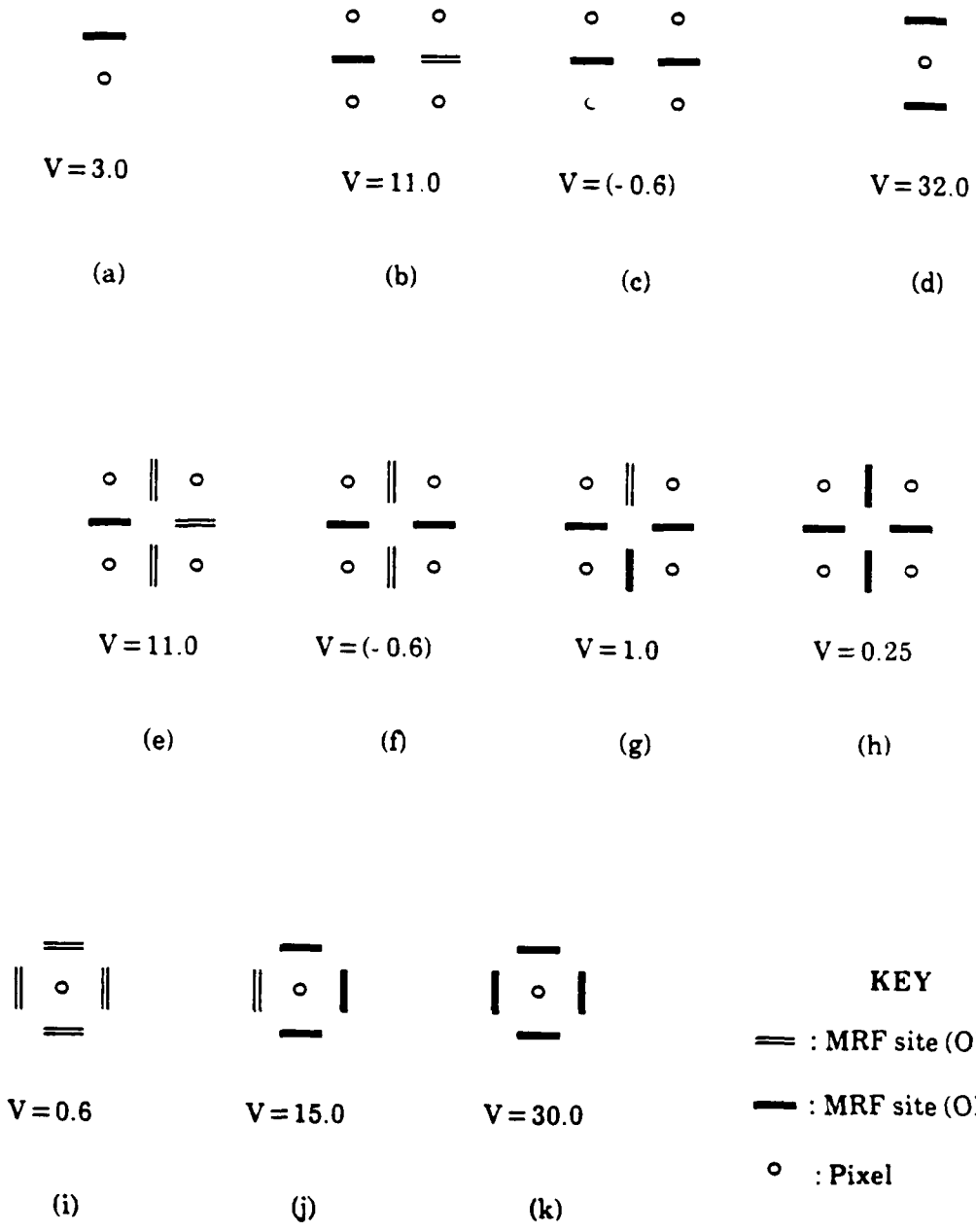


Figure 4: Potential Assignments for Cliques
(Configurations not shown have null (0.0)
potential values)

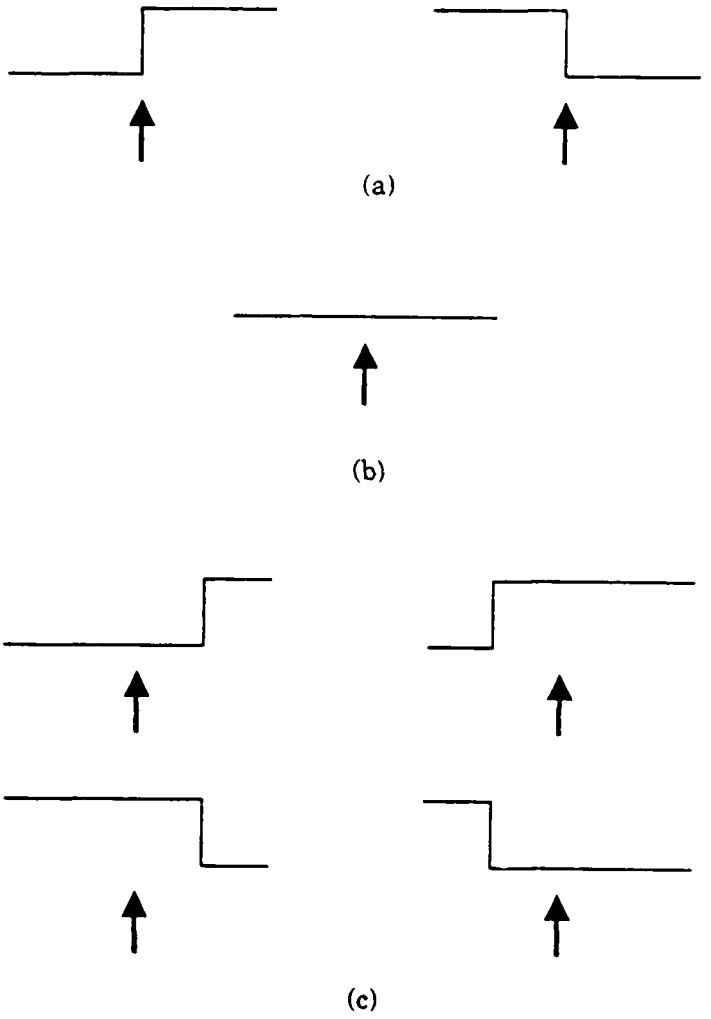


Figure 5: Image events in a 4X1 window
(a) Edge occuring at center of window,
(b) Homogenous region: no edge occurs,
(c) No edge at center, offset edge occurs.
(Arrow indicates center of window)

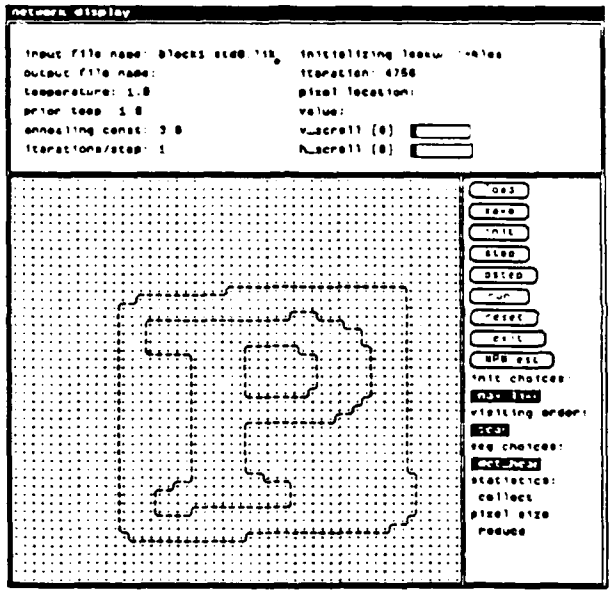


Figure 6: An interactive general purpose MRF simulator.

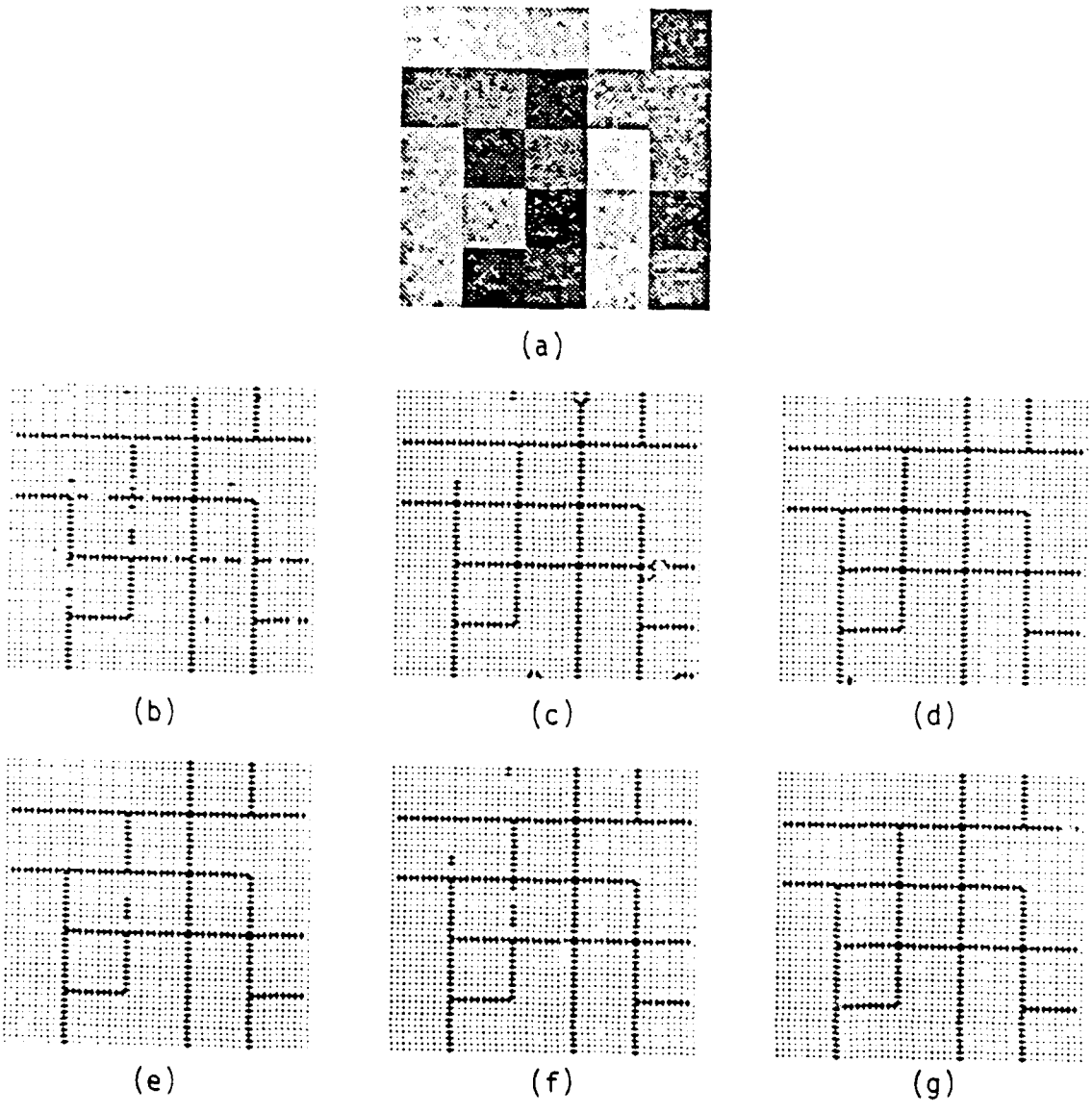


Figure 7: Experiment set. (a) Synthetic 50×50 "checkerboard" image corrupted by independent Gaussian noise, mean 0, standard deviation 16.0. (b) TLR configuration. (c) Stochastic MAP estimate. (d) Stochastic MPM estimate. (e) DIR (scan-line visiting order) MAP estimate. (f) DIR (random visiting order) MAP estimate. (g) HCF result.

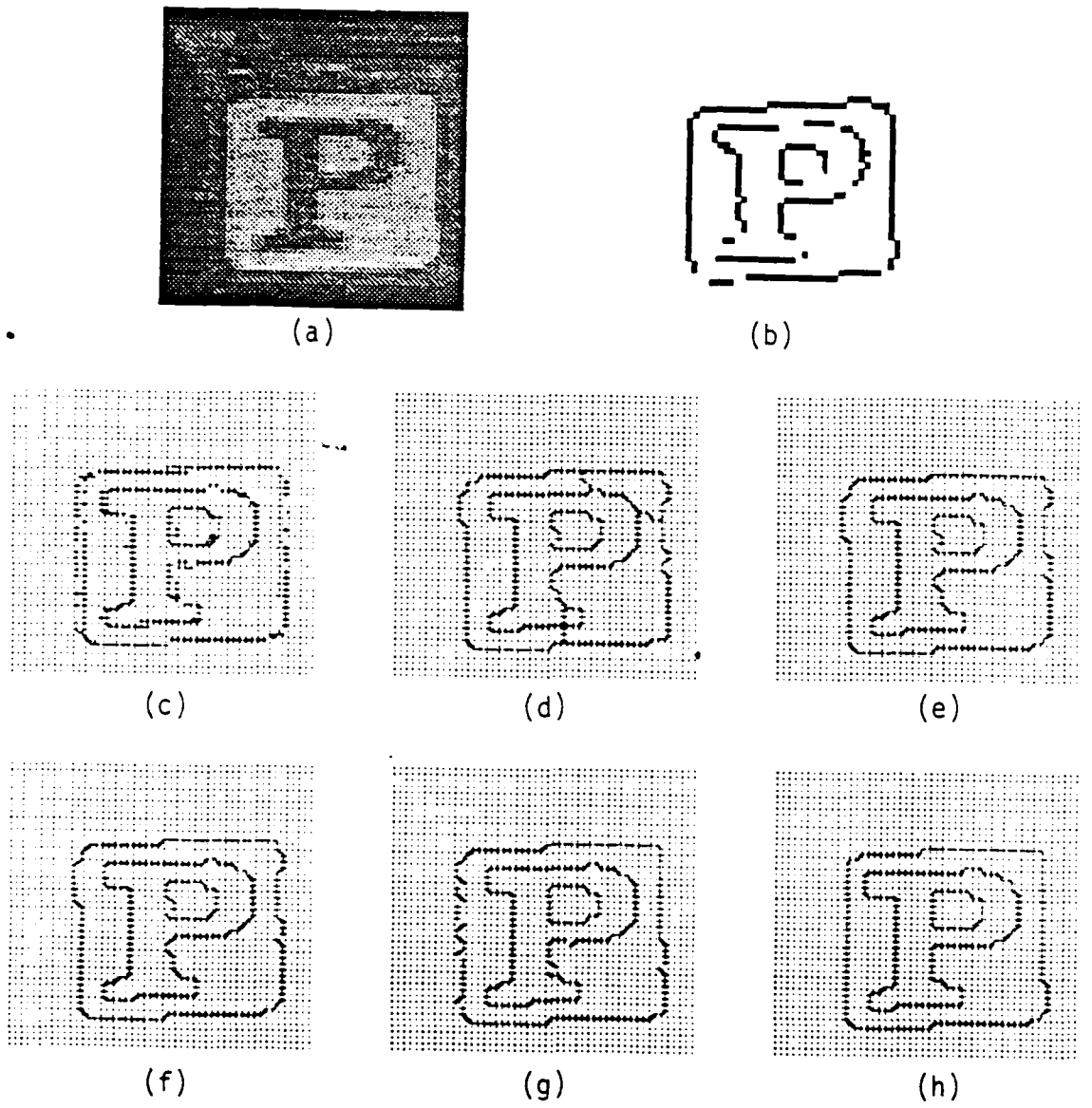


Figure 8: Experiment set. (a) Natural 50×50 image of wooden block. (b) Thinned and thresholded output of 3×3 Kirsch operators. (c) TLR configuration. (d) Stochastic MAP estimate. (e) Stochastic MPM estimate. (f) DIR (scan-line visiting order) MAP estimate. (g) DIR (random visiting order) MAP estimate. (h) HCF result.

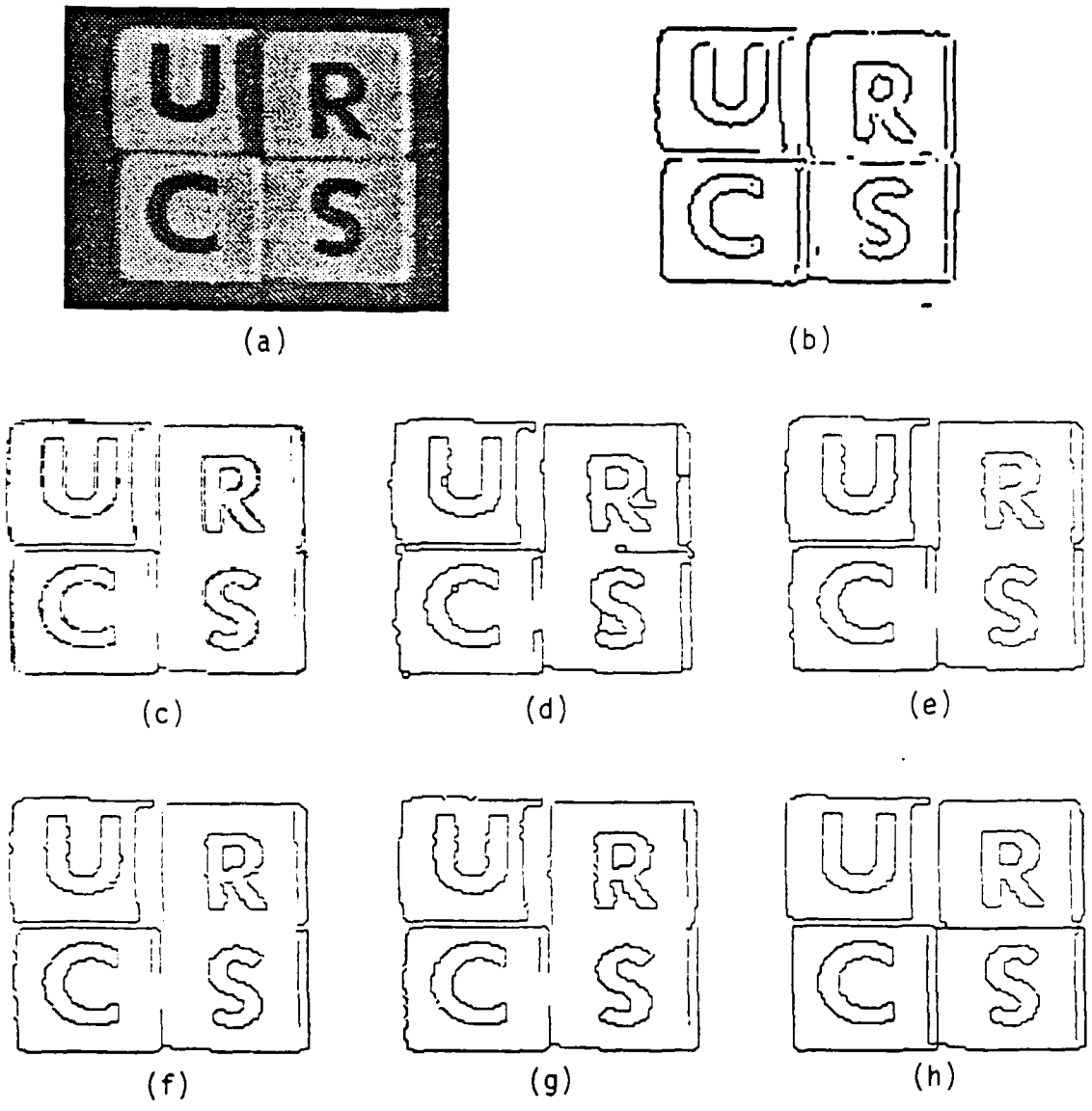


Figure 9: Experiment set. (a) Natural 100×124 image of four plastic blocks. (b) Thinned and thresholded output of 3×3 Kirsch operators. (c) TLR configuration. (d) Stochastic MAP estimate. (e) Stochastic MPM estimate. (f) DIR (scan-line visiting order) MAP estimate. (g) DIR (random visiting order) MAP estimate. (h) HCF result.

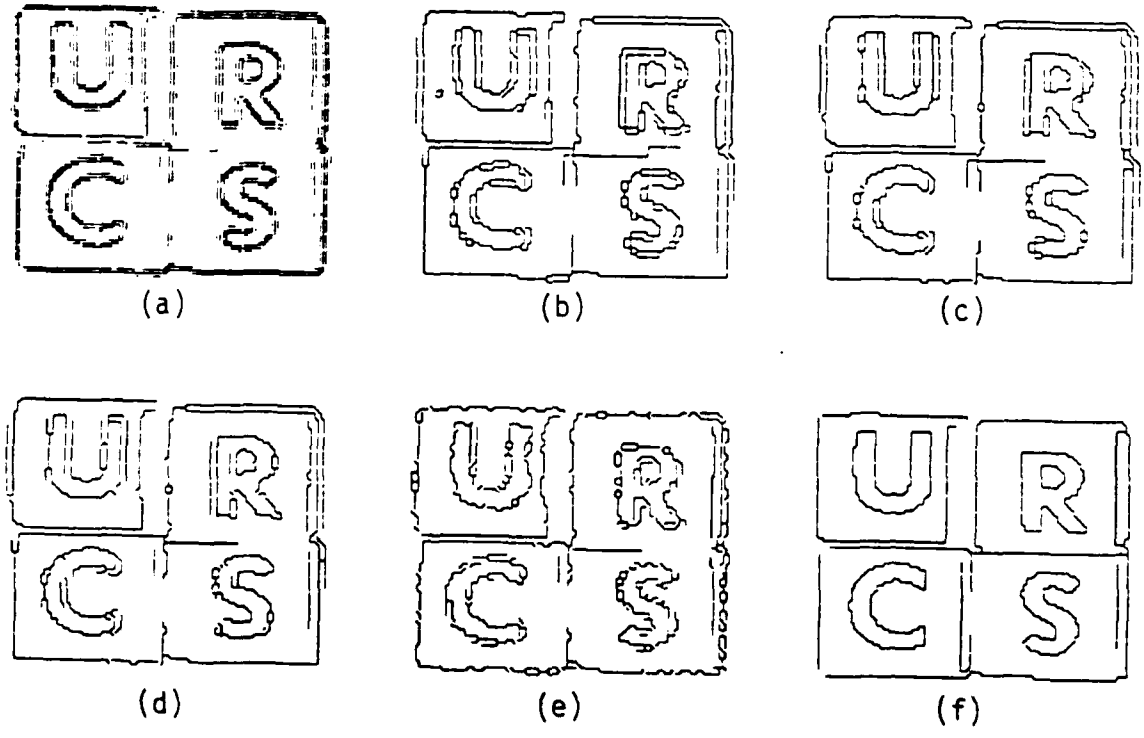


Figure 10: Experiments with incomplete edge model - original image in Fig. 9a. (a) TLR configuration. (b) Stochastic MAP estimate. (c) Stochastic MPM estimate. (d) DIR (scan-line visiting order) MAP estimate. (e) DIR (random visiting order) MAP estimate. (f) HCF result.

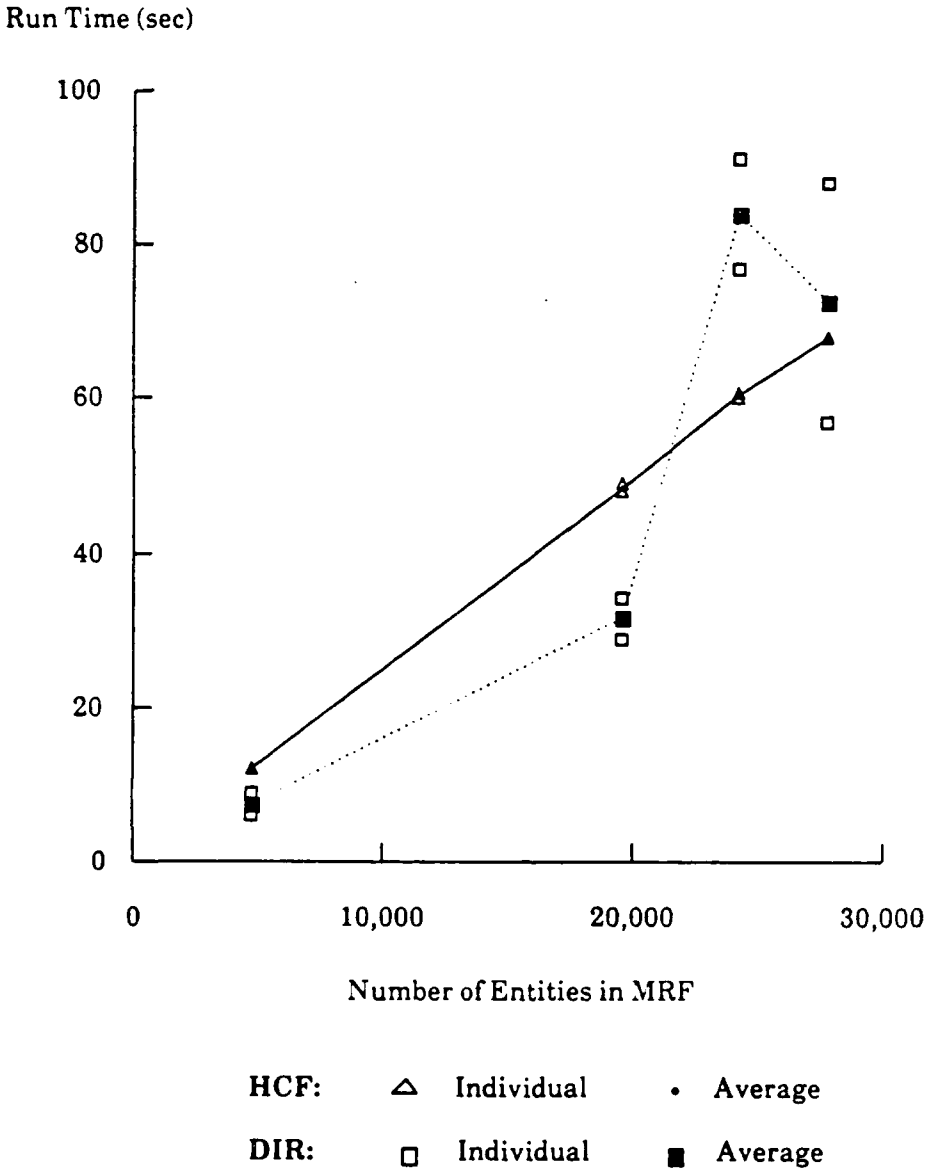


Table 1: Timing Test Results. The HCF and DIR algorithms are each run on two images of the same size, for four image sizes. Individual and average run-times are shown.